2020-12

USING MODEL-BASED SYSTEMS ENGINEERING METHODS TO CAPTURE A DEPARTMENT OF DEFENSE ACQUISITION LIFE CYCLE

Torok, Ronald J.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

USING MODEL-BASED SYSTEMS ENGINEERING METHODS TO CAPTURE A DEPARTMENT OF DEFENSE ACQUISITION LIFE CYCLE

by

Ronald J. Torok

October 2021

Thesis Advisor: Warren Vaneman
Second Reader: Clifford A. Whitcomb

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** October 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
USING MODEL-BASED SYSTEMS ENGINEERING METHODS TO CAPTURE A DEPARTMENT OF DEFENSE ACQUISITION LIFE CYCLE

5. FUNDING NUMBERS

6. AUTHOR(S) Ronald J. Torok

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD[SE]) is pushing model-based systems engineering (MBSE) methods to increase efficiencies and technical rigor in Department of Defense (DoD) engineering practices. MBSE methods might also aid in the planning of an acquisition process. An MBSE process is proposed for capturing the acquisition life cycle, the structure that implements the life cycle processes, and the developed information artifacts using a SysML model. The SE processes for an example middle tier of acquisition (MTA) program are modeled and model views of how program information, functions, and developed information artifacts are shown. The MBSE approach with the ability to synchronize information across views and use data queries to extract information into tables or matrixes was found to be beneficial for planning required acquisition life cycle functions, capturing the resources or informational elements the functions need, identifying a national team and required infrastructure, and allocating work breakdown structure (WBS) elements to those team members or infrastructure. The modeling approach allows additional flexibility for programs to extend or modify the processes to meet their specific needs. Additional work is needed to develop methods to simulate and analyze the model for benefits of alternative development approaches.

14. SUBJECT TERMS
Office of the Deputy Assistant Secretary of Defense for Systems Engineering, ODASD(SE), MBSE, digital engineering, Department of Defense, DOD, International Council on Systems Engineering, INCOSE, Object-Oriented Systems Engineering Method, OOSEM, middle tier of acquisition, MTA, model-based engineering, model-based systems engineering, systems engineering, SE, SysML, acquisition, WBS, work breakdown structure

15. NUMBER OF PAGES
115

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

USING MODEL-BASED SYSTEMS ENGINEERING METHODS TO CAPTURE A DEPARTMENT OF DEFENSE ACQUISITION LIFE CYCLE

Ronald J. Torok
Civilian, Department of the Navy
BS, Rose-Hulman Institute of Technology, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
October 2021

Approved by: Warren Vaneman
Advisor

Clifford A. Whitcomb
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD[SE]) is pushing model-based systems engineering (MBSE) methods to increase efficiencies and technical rigor in Department of Defense (DoD) engineering practices. MBSE methods might also aid in the planning of an acquisition process. An MBSE process is proposed for capturing the acquisition life cycle, the structure that implements the life cycle processes, and the developed information artifacts using a SysML model. The SE processes for an example middle tier of acquisition (MTA) program are modeled and model views of how program information, functions, and developed information artifacts are shown. The MBSE approach with the ability to synchronize information across views and use data queries to extract information into tables or matrixes was found to be beneficial for planning required acquisition life cycle functions, capturing the resources or informational elements the functions need, identifying a national team and required infrastructure, and allocating work breakdown structure (WBS) elements to those team members or infrastructure. The modeling approach allows additional flexibility for programs to extend or modify the processes to meet their specific needs. Additional work is needed to develop methods to simulate and analyze the model for benefits of alternative development approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- **A. MOTIVATION....2**
- **B. RESEARCH APPROACH....4**
- **C. THESIS ORGANIZATION....6**

II. LITERATURE REVIEW....9
- **A. SYSTEMS ENGINEERING LIFE CYCLE REFERENCE MODELS....9**
  - **1. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*....10**
  - **2. Defense Acquisition Guidebook....13**
- **B. MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE MODELS....17**
  - **1. Object-Oriented Systems Engineering Method....17**
  - **2. "Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions" (Stepanchick 2016)....21**

III. MODELING A MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE PROCESSES....25
- **A. MBSE FRAMEWORK MODEL PLAN....28**
- **B. MBSE FRAMEWORK STAKEHOLDER NEEDS AND PROCESS REQUIREMENTS....30**
- **C. MBSE FRAMEWORK REQUIREMENT CAPTURE....33**
- **D. MBSE FRAMEWORK LOGICAL ARCHITECTURE....35**
- **E. MBSE FRAMEWORK PROCESS....37**
- **F. EVALUATE AND MAINTAIN MBSE FRAMEWORK MODEL....39**

IV. MODELING A MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE PROCESS....41
- **A. MBSE FRAMEWORK MODEL PLAN....42**
- **B. MBSE FRAMEWORK STAKEHOLDER NEEDS AND PROCESS REQUIREMENTS....44**
- **C. MBSE FRAMEWORK REQUIREMENT CAPTURE....49**
- **D. MBSE FRAMEWORK LOGICAL ARCHITECTURE....53**
  - **1. MBSE Framework Logical Structure....53**
  - **2. MBSE Framework Logical Behavior....55**

3. MBSE Framework Logical Behavior Mapping to Current Process Behavior 64
4. MBSE Framework Logical Interfaces 66
5. MBSE Framework Logical Digital System Model....67
E. MBSE FRAMEWORK IMPLEMENTATION 68
1. MBSE Framework Logical to Implementation Mapping Criteria....69
2. MBSE Framework Implementation Structure 69
3. MBSE Framework Implementation Behavior 70
4. MBSE Framework Implementation to Logical Mapping 73
5. MBSE Framework Implementation Interfaces....75
6. Implementation Digital System Model....76
F. EVALUATE AND MAINTAIN MBSE FRAMEWORK MODEL 77

V. CONCLUSIONS 83

LIST OF REFERENCES 87

INITIAL DISTRIBUTION LIST 89

LIST OF FIGURES

Figure 1. United States Navy and People's Liberation Army Navy Capability Fielding. Source: Grosklags (2017). 3

Figure 2. ISO Life cycle Stages. Source: INCOSE (2015, 29). 10

Figure 3. Life cycle Processes. Source: INCOSE (2015, 2)....11

Figure 4. *INCOSE Systems Engineering Handbook* Architecture Definition Process. Source: INCOSE (2015, 65). 12

Figure 5. INCOSE Systems Engineering Process Tailoring. Source: INCOSE (2015, 146). 13

Figure 6. DoD Weapon System Development Life Cycle. Source: DoD (2018a). 14

Figure 7. Defense Unique Software Intensive Program Acquisition Model. Source: DoD (2017). 15

Figure 8. OOSEM Recursive Process. Source: Friedenthal, Moore, and Steiner (2014, 420). 18

Figure 9. OOSEM Activities. Source: Friedenthal, Moore, and Steiner (2014, 423). 19

Figure 10. MBSE Integration into Acquisition Life cycle. Source: Stepanchick (2016). 23

Figure 11. MBSE Framework Development Process 25

Figure 12. DAG Systems Engineering Process to OOSEM Process 27

Figure 13. Example Package Structure. Source: Friedenthal, Moore, and Steiner (2014, 430)....29

Figure 14. Analyze Stakeholder Needs for MBSE Framework 30

Figure 15. Example Framework Use Case Diagram....32

Figure 16. Relating Capabilities to Use Cases Example....32

Figure 17. Analyze MBSE Framework Requirements....33

Figure 18. MBSE Framework Use Cases to Scenario Relationships....34

Figure 19. Example State Diagram 35
Figure 20. Define Logical MBSE Framework Architecture 35
Figure 21. Define the MBSE Framework Architecture 37
Figure 22. Implementation to Work Breakdown Structure Allocation Metamodel 38
Figure 23. Adaptive Acquisition Framework. Source: DoD (2019). 42
Figure 24. Modeling Guidelines Structure 43
Figure 25. Missile Development Systems Engineering Life Cycle Model Package Structure....44
Figure 26. MBSE Framework Stakeholders....45
Figure 27. Acquisition Domain 47
Figure 28. MBSE Framework Use Cases....48
Figure 29. Capabilities to Encourage Artifact Reuse 48
Figure 30. MBSE Framework Scenario for Interactions between OSD and MBSE Framework 50
Figure 31. MBSE Framework Context Diagram....51
Figure 32. MTA Development States....52
Figure 33. MBSE Framework Work Breakdown Structure 54
Figure 34. Activity Design Architecture with Logical Allocations 57
Figure 35. Artifact Taxonomy for Interfaces 59
Figure 36. SysML Action Polymorphism with Manage Configuration Example....59
Figure 37. Activity Design Digital System with Logical Allocations 61
Figure 38. Activity Verify System with Logical Allocations 62
Figure 39. Technical Data Package Elements for Input to Develop Functional Design 63
Figure 40. Mapping of Logical Architecture Process to INCOSE SE Processes. Adapted from INCOSE (2015). 65

Figure 41. Allocation of Digital Taxonomy 68
Figure 42. MBSE Framework 70
Figure 43. Program Office Decomposition 71
Figure 44. Activity Verify System with Implementation Allocations 72
Figure 45. MBSE Framework Implementation to Logical Structure Mapping 74
Figure 46. Ontology for Systems Engineering Plan 77
Figure 47. Cameo Structured Expression 78
Figure 48. Validation Script Constraint 79
Figure 49. Structured Expression to Identified Actions Not Allocated to WBS 80
Figure 50. Structured Expression for Abstraction Relationship between Work Breakdown Structure and Implementation Structure 81
Figure 51. Structured Expression to Identify Interfaces between Work Breakdown Structure Parts 82

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. DAG SE Processes Mapped to ISO 15288 SE Processes. Source: DoD (2018a). 16

Table 2. Example Process Modeling Guidelines 29

Table 3. Example Logical Function Information 36

Table 4. MBSE Framework Stakeholder Needs 46

Table 5. MBSE Framework Example Requirements 53

Table 6. Technical Data Package Elements for Input to Develop Functional Design 64

Table 7. INCOSE Documentation for Design Architecture Action. Adapted from INCOSE (2015, 59, 65–67). 66

Table 8. Logical Architecture Allocation Example for Develop Architecture 67

Table 9. Implementation Architecture Allocation Example for Develop Architecture 75

Table 10. Prime Contractor Developed Artifacts 76

Table 11. Validation Script Output 80

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	application programming interfaces
CDD	Capability Development Document
CDR	Critical Design Review
CFD	computational fluid dynamics
DAG	*Defense Acquisition Guidebook*
DoD	Department of Defense
DSM	digital system model
HWIL	hardware in the loop
ibd	internal block diagram
INCOSE	International Council on Systems Engineering
ICD	Initial Capabilities Document
ISO	International Organization for Standardization
IT	information technology
JCIDS	Joint Capabilities Integration and Development System
LCDR	Lieutenant Commander
MBSE	model-based systems engineering
MBE	model-based engineering
MCA	major capability acquisition
MOE	measure of effectiveness
MTA	middle tier of acquisition
NASA	National Aeronautics and Space Administration
NDIA	National Defense Industrial Association
NSERC	Naval Systems Engineering Research Center
ODASD(SE)	Office of the Deputy Assistant Secretary of Defense for Systems Engineering
OOSEM	Object-Oriented Systems Engineering Method
OQE	objective quality evidence
OSD	Office of the Secretary of Defense
PDR	Preliminary Design Review
PLA(N)	People’s Liberation Army Navy

PLM	Product Lifecycle Manager
RAM	reliability, availability, and maintainability
SCG	Security Classification Guide
SE	systems engineering
SEP	Systems Engineering Plan
SETR	Systems Engineering Technical Review
SFR	System Functional Review
SRR	System Requirements Review
SWIL	software in the loop
SysML	Systems Modeling Language
TDP	technical data package
TRAP	Technical Review Action Plan
WBS	work breakdown structure
UML	Unified Modeling Language
USN	United States Navy
VADM	Vice Admiral

EXECUTIVE SUMMARY

A. MOTIVATION FOR RESEARCH

Department of Defense (DoD) military systems are becoming increasingly more complex, and traditional systems engineering (SE) processes need to be modernized to handle the complexity of both the design of a system and the analysis conducted on that design. Model-based systems engineering (MBSE) is one method that has been prescribed by several organizations to handle this complexity including the National Defense Industrial Association (NDIA). The NDIA defines MBSE as "an approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle" (National Defense Industrial Association [NDIA] 2011). For the purpose of this thesis, MBSE is used in place of other popular terms such as model-based engineering (MBE) or Digital Engineering as a catch-all term for any engineering that is conducted with a model-based approach.

The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD(SE)) defined the term Digital Engineering to broaden the scope of MBSE. The ODASD(SE) (2018) defines Digital Engineering as "An integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life cycle activities from concept through disposal." A Digital Engineering Ecosystem is an enabler for Digital Engineering defined by the ODASD(SE) (2018) as "The interconnected infrastructure, environment, and methodology (process, methods, and tools) used to store, access, analyze, and visualize evolving systems' data and models to address the needs of the stakeholders." MBSE framework will be used to describe methods, process, infrastructure, and the digital artifacts used to conduct MBSE during the acquisition life cycle.

B. RESEARCH QUESTIONS

With the motivation in mind, the thesis answers the following questions:

1. How can an MBSE approach be used to define the systems engineering process for the acquisition life cycle?
2. How can MBSE methods be inserted into a model of the systems engineering process for the acquisition life cycle?

C. REFERENCE PROCESSES

Traditional systems engineering can be a bridge between document-based methods and MBSE methods. MBSE must meet the objects of traditional systems engineering while adding new methods. The International Council on Systems Engineering's (INCOSE) *Systems Engineering Handbook* expands upon the industry standard ISO 15288 and provides a foundation for how systems engineering should be implemented (International Council on Systems Engineering [INCOSE] 2015). *The Defense Acquisition Guidebook* (DAG) supplies a more acquisition focused approach that better aligns to the DoD's needs (DoD 2018a). By reviewing traditional systems engineering methods, a checklist can be formed for much of what MBSE must accomplish. The SE community has documented many methods for engineers to implement MBSE. The Object-Oriented Systems Engineering Method (OOSEM) provides robust methods for capturing and managing system requirements and architecture well, especially with using an object-oriented architecture language like SysML. However, OOSEM lacks details for how systems engineers should use analytical models. A thesis by Lieutenant Commander (LCDR) Stepanchick, "Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions" (2016), provides insight for how Navy developers can benefit from and mature all types of models across the life cycle with from a Navy perspective.

D. PROCESS TO MODEL AN ACQUISTION LIFE CYCLE

Figure 1 shows a process for the life cycle definition team to define an MBSE framework. The process is a modification of the OOSEM process for developing a system from *A Practical Guide to SysML* (Friedenthal, Moore, and Steiner 2014, 417–503). The process shown is a linear process, but modelers should use iteration as they require.

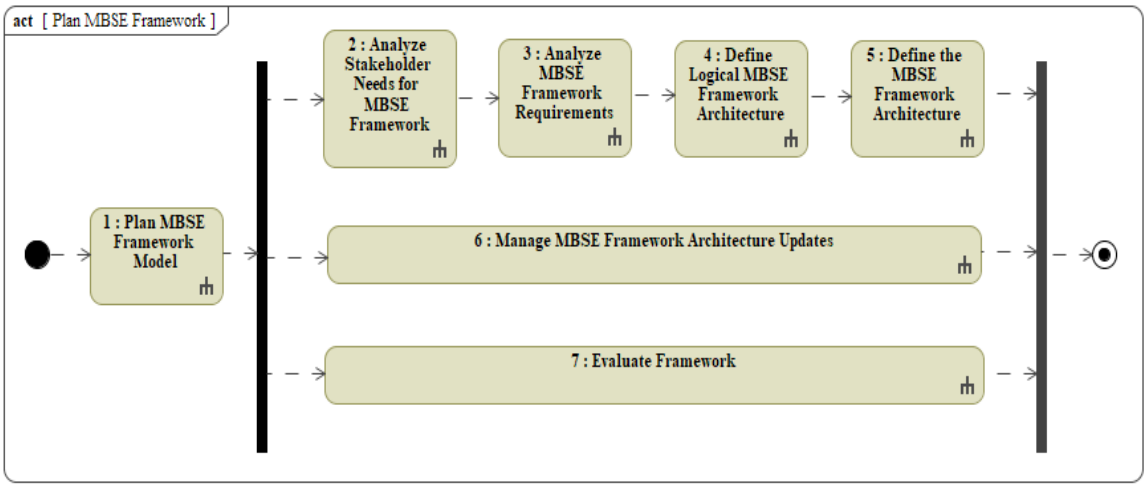

Figure 1. MBSE Framework Development Process

An architecture model can aid an organization in similar ways as a system architecture model would. The model can trace decisions in the design of the life cycle model to requirements, rationale, or stakeholder needs to understand the importance of acquisition functions. Interrelationships between items, structure, and behavior can trace in multiple dimensions to allow users of the process to view different aspects based on their viewpoint of the process. A test engineer has different needs from the process than a logistics engineer, and a model can assist in creating these alternate views while maintaining integration between the views.

E. EXAMPLE MODEL AND CONCLUSIONS

The project modeled a middle tier of acquisition (MTA) (Section 804) program to allow flexibility in defining the systems engineering process to be more MBSE focused. Examples for how to capture traditional information within a process such as a WBS, responsibilities for program office vs contractor, and the information needs between them was shown. Manual built relationships between elements were shown in addition to examples for relationships that can be queried from the model.

The greatest benefit of modeling the acquisition process was found to the ability to look at the captured data from multiple views. Tables and matrices that are automatically updated after the queries are created were highly beneficial in understanding the content in the model. The model also allowed for the ability to reference traditional processes more specifically, or current program modeled processes more finely as opposed to the general method to reference an entire document that may be hundreds of pages. The model can also then be extended and referenced by others as they tailor the process to create an acquisition framework for a program's specific needs.

The second question concerned modeling MBSE methods within the process. MBSE processes were reviewed and the best practices identified were incorporated into the MTA acquisition process modeled. The ability to substitute new SysML Activities was found to be beneficial for modifying processes to incorporate MBSE methods. MBSE methods in general do not alter the overall traditional SE processes, but the more detailed functions below do need to be updated with different artifact inputs and outputs.

The model was also beneficial in defining the artifacts of an MBSE process. In traditional paper-based methods, some form of a document or drawing is the information being exchanged and the source of truth for information. In MBSE methods, the digital artifacts that can be manipulated, queried, and/or executed should be the artifacts based. The model was beneficial in looking at a taxonomy and structure for these elements and then mapping this richer data to the processes used to create and input them.

References

Department of Defense. 2018a. "*Defense Acquisition Guidebook*." February 26. Washington, DC: Department of Defense. Retrieved from https://www.dau.mil/tools/dag.

Friedenthal, Sanford, Rick Steiner, and Alan Moore. 2014. *A Practical Guide to SysML*, 3rd ed. Burlington, MA: Morgan Kaufmann.

International Council on Systems Engineering. 2015. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th ed. edited by David Walden, Garry Roedler, Kevin Forsberg, Douglas Hamelin, and Thomas Shortell. Hoboken: Wiley.

NDIA Systems Engineering Division M&S Committee. (2011). Final Report of the Model Based Engineering (MBE) Subcommittee. Retrieved from https://www.ndia.org/-/media/sites/ndia/meetings-and-events/3187-sullivan/divisions/systems-engineering/modeling-and-simulation/reports/model-based-engineering.ashx

Office of the Deputy Assistant Security of Defense for Systems Engineering ODASD(SE). (n.d.). DoD Systems Engineering - Initiatives. Retrieved January 15, 2018, from https://www.acq.osd.mil/se/initiatives/init_de.html

Stepanchick, Justin. 2016. "Integrating Model Based Engineering and Trade Space Exploration into naval acquisitions" Master's thesis, Massachusetts Institute of Technology. http://dspace.mit.edu/handle/1721.1/104297.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I'm extremely grateful to have worked with my thesis advisor, Dr. Warren Vaneman. He stuck with me even though this thesis took longer than expected and helped drive the research to what would be beneficial to both the MBSE, academic, and acquisition communities.

Special thanks to work colleagues that provided feedback to the model methodology, helped create reference models for INCOSE or DAG processes, or helped in the generation of the example model:

Jacob Chapman

Tom Payne

Josh Hanback

Kim Sommer

Patrick Finn

Many thanks to NSWC Crane for supporting me in the program. This was a great opportunity to expand myself as a systems engineer.

I also want to thank Heather Hahn, who always reminded me of deadlines and worked with me to meet those deadlines.

I'd like to acknowledge Susan Hawthorne, who did a great job cleaning up my writing and ensuring my thesis was formatted correctly.

Finally, I want to thank my family for their support. They put up with me working long weekends and nights to finalize this thesis in the middle of a pandemic!

THIS PAGE INTENTIONALLY

I. INTRODUCTION

Department of Defense (DoD) military systems are becoming increasingly more complex, and traditional systems engineering (SE) processes need to be modernized to handle the complexity of both the design of a system and the analysis conducted on that design. Model-based systems engineering (MBSE) is one method that has been prescribed by several organizations to handle this complexity including the National Defense Industrial Association (NDIA). The NDIA defines MBSE as "an approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle" (National Defense Industrial Association [NDIA] 2011). For the purpose of this thesis, MBSE is used in place of other popular terms such as model-based engineering (MBE) or digital engineering as a catch all term for any engineering that is conducted with a model-based approach.

The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD(SE)) defined the term Digital Engineering to broaden the scope of MBSE. The ODASD(SE) (2018) defines Digital Engineering as "An integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life cycle activities from concept through disposal." A Digital Engineering Ecosystem is an enabler for Digital Engineering defined by the ODASD(SE) (2018) as "The interconnected infrastructure, environment, and methodology (process, methods, and tools) used to store, access, analyze, and visualize evolving systems' data and models to address the needs of the stakeholders." MBSE framework will be used to describe methods, process, infrastructure, and the digital artifacts used to conduct MBSE during the acquisition life cycle.

This thesis inspects the DoD acquisition life cycle for where MBSE can provide the most impact. The processes used to develop and manage a product's life cycle are broad, and some of these processes inherently lend themselves to an MBSE approach. Some processes should be supported by MBSE, whereas others will have little connection to an MBSE framework. The majority of the current systems engineering and DoD

acquisition process definition is also document-based. If an MBSE approach is beneficial for developing a system, MBSE should also be able to aid defining the framework in which that system is developed. This paper will show a method for defining a process using MBSE methods while still showing linkage to traditional systems engineering process documentation.

A. MOTIVATION

VADM Grosklags, at the 2017 NDIA Systems Engineering Conference, spoke of the need for the Navy to increase the speed for providing capabilities to the Warfighter. Figure 1 was presented by VADM Grosklags, and the figure shows speed at which the People's Liberation Army Navy (PLA(N)) has fielded new capabilities in comparison to the United States Navy (USN). The Navy needs to speed the fielding of new capabilities to the Warfighter in order to maintain technological advantage. Current practices across the life cycle acquisition need to have increased efficiency in order to compete. VADM Grosklags identified MBSE as a potential solution for increasing efficiency in some DoD acquisition practices (Grosklags 2017).

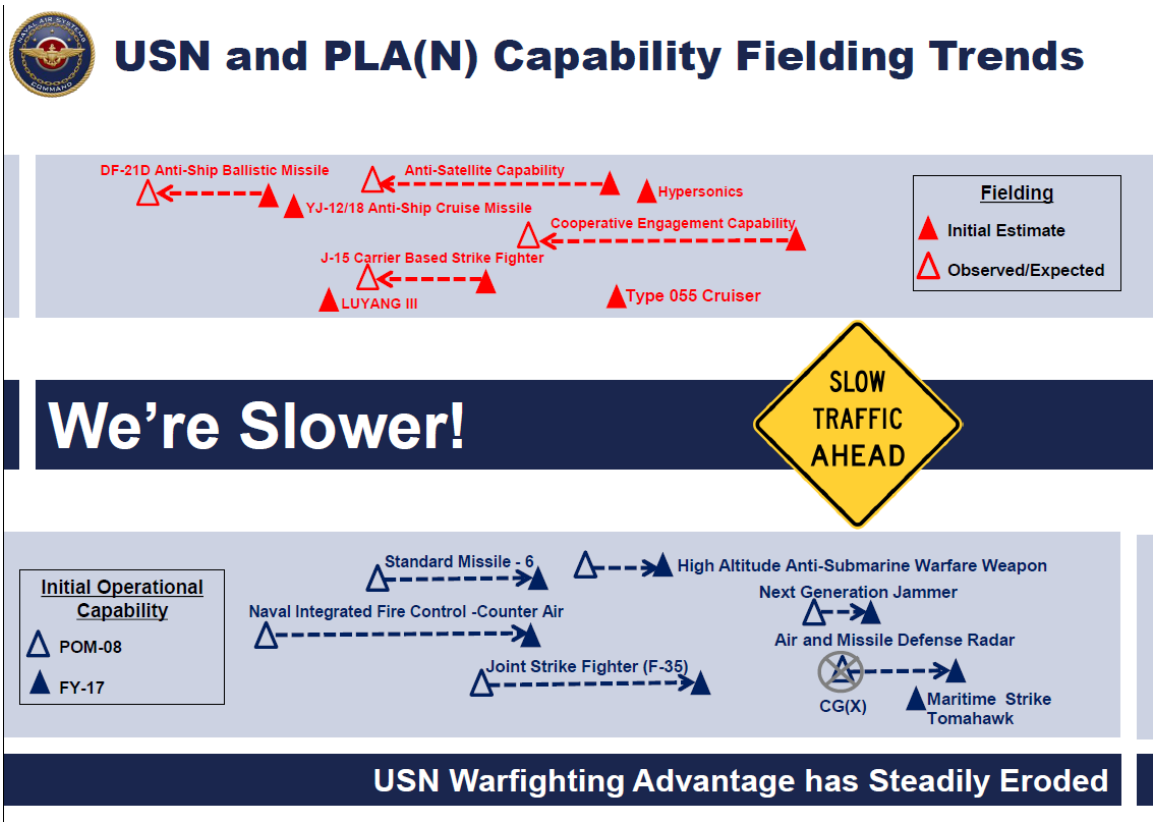

Figure 1. United States Navy and People's Liberation Army Navy Capability Fielding. Source: Grosklags (2017).

However, DoD process guidance has not articulated well where MBSE methods should be employed, what acquisition document-based products can be replaced or supported by MBSE artifacts, or how information should be captured in these products. The DoD acquisition process as defined in DoD Instruction (DoDI) 5000.02 (Department of Defense [DoD] 2017) and the *Defense Acquisition Guidebook* (DoD 2018a) are broader than even the industry processes as defined in the *INCOSE Systems Engineering Handbook (International Council on Systems Engineering [INCOSE] 2015).* Many systems engineering processes lend themselves well to an MBSE approach while others can be supported by an MBSE approach. Most important, a one-size fits all solution is not appropriate. System definition and analysis needs to be conducted based on the needs of the project. Some projects may require a detailed reliability, availability, and maintainability (RAM) analysis to be able to sustain a long operational life cycle. Other DoD projects may require a quick response to fielding in which long- term fielding a rich

RAM analysis is not required for the first deployment of the capability. Modern life cycle acquisition process requires flexibility, and an acquisition process definition methodology must facilitate that flexibility.

One area of concern that MBSE may be able to help with is reducing errors inserted in systems engineering artifacts across the life cycle. As a system definition matures, developers inevitably insert errors as they capture system information. These errors can be in the form of misaligned requirements, poor assumptions, incorrect analytical models, or a large number of other sources. A system's engineer's goal should be to identify these errors as quickly as possible to resolve them earlier in the life cycle, and MBSE framework should facilitate preventing these errors and finding them quickly. Finding errors early can dramatically reduce the cost to take correct action. NASA found the cost to fix errors rise dramatically as the life cycle going from a 1x factor during the requirements phase, to a 16x factor during manufacturing, to a 29x factor during production (Haskins et al. 2004). These errors are also causing schedule delays to make the correction. These schedule delays are leading to capabilities being delivered late to the Warfighter. MBSE methods for the acquisition process must attempt to reduce these errors.

With the motivation in mind, this thesis answers the following questions:

1. How can an MBSE approach be used to define the systems engineering process for the acquisition life cycle?
2. How can MBSE methods be inserted into a model of the systems engineering process for the acquisition life cycle?

B. RESEARCH APPROACH

The research focus is to identify MBSE methods to define systems engineering activities in an acquisition lifecycle by modeling the acquisition process. MBSE methods are being widely advertised as superior approaches to defining a system, but they should also benefit the definition of an acquisition process in which the outputs are complex set of plans, procedures, design artifacts, test and production elements, and finally a fielded weapon system. Examples for the use of and best practices for MBSE methods are available

for defining a system. A similar example for defining the acquisition process will be explored. By modeling the process, other programs or research should be able to extend the model to fit the needs of their program in a similar way that current document-based processes are tailored for a program. Benefits of tailoring a model over a tailoring a document-based product will be explored.

Several resources are available in paper-based format for guidance in how an acquisition program can be executed. These resources all rely on abstract descriptions of the process and varying levels of fidelity to describe the outputs from each step in the process. These elements will need to be mapped to MBSE modeling elements to MBSE modeling elements. The Systems Modeling Language (SysML) will be explored for how these elements can be defined in a model.

The second focus of the research focuses on how an MBSE framework can be defined for a project using an MBSE approach. This will start with identifying the digital the functions executed for an MBSE framework to replace or supplement current SE functions. One concern is tracing new MBSE methods and activities to traditional systems engineering activities in existing practice. System developers and managers of development are accustomed to current practices that are primarily document-based and follow long- standing systems engineering standards. To migrate a workforce to an MBSE framework, the MBSE processes must link back to the current processes they are replacing or supplementing.

The research approach to define the management of the digital system model (DSM) will be to use popular and regulatory artifacts for the SE process to define the scope of products. A method for describing the taxonomy of the artifacts and the information that they contain will be presented. The taxonomy will be used to describe the information contained within the artifacts. Lastly, a method for showing how information is expected to flow between the artifacts and the necessary interface definitions will be presented.

The final concern for both thesis questions is answering the motivation for using MBSE methods in the acquisition life cycle. Reducing errors in system definition across the life cycle is a top concern, and methods for reducing the insertion of errors or for

identifying these errors must be concerned as an MBSE process is developed. A focus when presenting these methods will be to follow some best practice recommended from Object Oriented Systems Engineering Methodology (OOSEM) and the thesis "Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions" (Stepanchick 2016).

C. THESIS ORGANIZATION

The second Chapter of this thesis is a literature review of the current state of systems engineering practices that were used for the research. This chapter will start with defining MBSE nomenclature used for this thesis and how that nomenclature relates to related terminology used by other organizations. Current systems engineering practices reviewed include the state of the DoD mandates for systems engineering during the acquisition process, standards used systems engineering, and methods to implement MBSE. The reference documents will also be reviewed for applicability to being presented in an MBSE approach. Resources for best practices in MBSE methods will be reviewed for their reuse in the example MBSE process defined in the third chapter of this thesis. Lastly, this chapter will also cover MBSE toolsets used for the research.

The third chapter of the thesis will cover the research to provide an MBSE approach for defining and managing systems engineering during the development of a system. This includes modeling a functional architecture for an MBSE process and a conceptual data architecture for the proposed set of artifacts and information created and captured during a DoD acquisition life cycle. The research will also provide methods for providing traceability to industry, government, and enterprise systems engineering processes. The MBSE approach for defining a life cycle process will provide extendibility and tailoring for future use and refinement.

The fourth chapter of this paper will demonstrate a tailoring of the provided MBSE process for use on a new missile system. Aspects of stakeholder needs will drive tailoring of the systems engineering processes for specific project needs. This will include providing some aspects of the missile as build to design, desirability for spiral development of some system capabilities, and requirements to field initial capability quickly.

The fifth chapter will make conclusions on the research conducted and presented in the thesis. It will also make recommendations for additional research to expand upon the work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

Traditional systems engineering can be a bridge between document-based methods and MBSE methods. MBSE must meet the objects of traditional systems engineering while adding new methods. The *INCOSE Systems Engineering Handbook* expands upon the industry standard ISO 15288 and provides a foundation for how systems engineering should be implemented (INCOSE 2015). *The Defense Acquisition Guidebook* (DAG) supplies a more acquisition focused approach that better aligns to the DoD's needs (DoD 2018a). By reviewing traditional systems engineering methods, a checklist can be formed for much of what MBSE must accomplish. The SE community has documented many methods for engineers to implement MBSE. The Object-Oriented Systems Engineering Method (OOSEM) provides robust methods for capturing and managing system requirements and architecture well, especially with using an object-oriented architecture language like SysML. However, OOSEM lacks details for how systems engineers should use analytical models. A thesis by Lieutenant Commander (LCDR) Stepanchick, "Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions" (2016), provides insight for how Navy developers can benefit from and mature all types of models across the life cycle with from a Navy perspective.

A. SYSTEMS ENGINEERING LIFE CYCLE REFERENCE MODELS

SE life cycle models represent approaches to developing a system from the identification of the need to the disposal of a system. SE life cycle models from standards bodies such as the International Organization for Standardization (ISO) and INCOSE offer a viewpoint applicable across most development projects. Federal organization SE life cycle models, such as those from the DoD and National Aeronautics and Space Administration (NASA), differ by focusing more heavily on the acquisition aspects of systems the federal government contracts out a larger percentage of the development and operations under federal regulations.

Several specialized SE life cycle models also exist which may be more focused on an aspect of a system or type of system. Life cycle models exist for various approaches to software development, the use of MBSE during system development, or the development and operations of a manufacturing capability. Even within the DoD, several example life cycles exist with the expectation projects will further refine these examples for their system's needs.

1. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*

The *INCOSE Systems Engineering Handbook* gives an overview of the life cycle of a system with a breakdown of phases as Concept, Development, Production, Utilization, Support, and Retirement as referenced from ISO 15288 and ISO 24748-1 as seen in Figure 2 (INCOSE 2015, 29). These system life cycle concepts will not change with MBSE, so an MBSE approach must adhere to these existing stages.

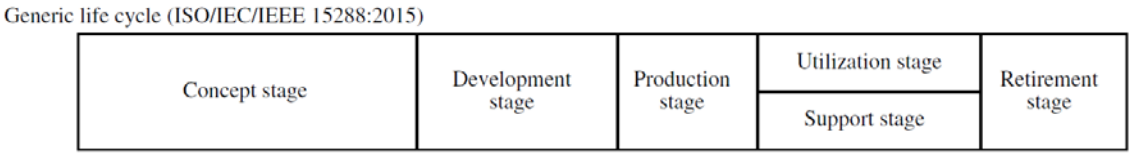

Figure 2. ISO Life cycle Stages. Source: INCOSE (2015, 29).

The handbook gives guidance on implementing the systems engineering processes seen in Figure 3. The handbook begins with technical processes that begin by analyzing an organization's business or mission to identify the need for a system at the beginning of a system's life cycle to the disposal process at the end of a system's life cycle. Figure 4 provides an example for the architecture definition processes and work products MBSE will need to provide. The technical management processes deal with managing the process, decisions, risks, and the data required across the system's life cycle. Technical Management processes will need to change with MBSE methods by being more agile and by benefiting from the rich data the models contain. MBSE will even affect the agreement processes related to acquiring portions of a system or supplies required for a system

because the acquisition of models with physical products can benefit an acquirer's understanding of acquired systems or subsystems. The organizational project-enabling processes aid not just the development of a single system, but the development of all systems within an organization. As organizations institutionalize MBSE methods, MBSE will provide requirements for or provide benefit to many of the organizational project-enabling processes.

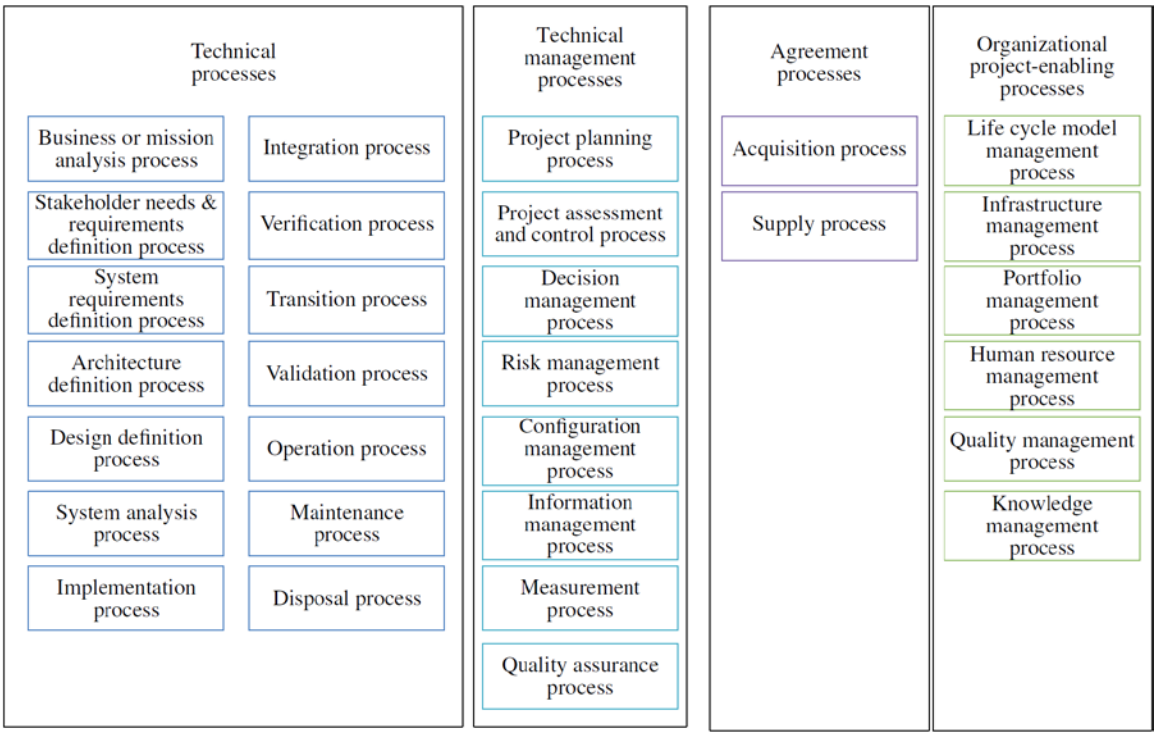

Figure 3. Life cycle Processes. Source: INCOSE (2015, 2).

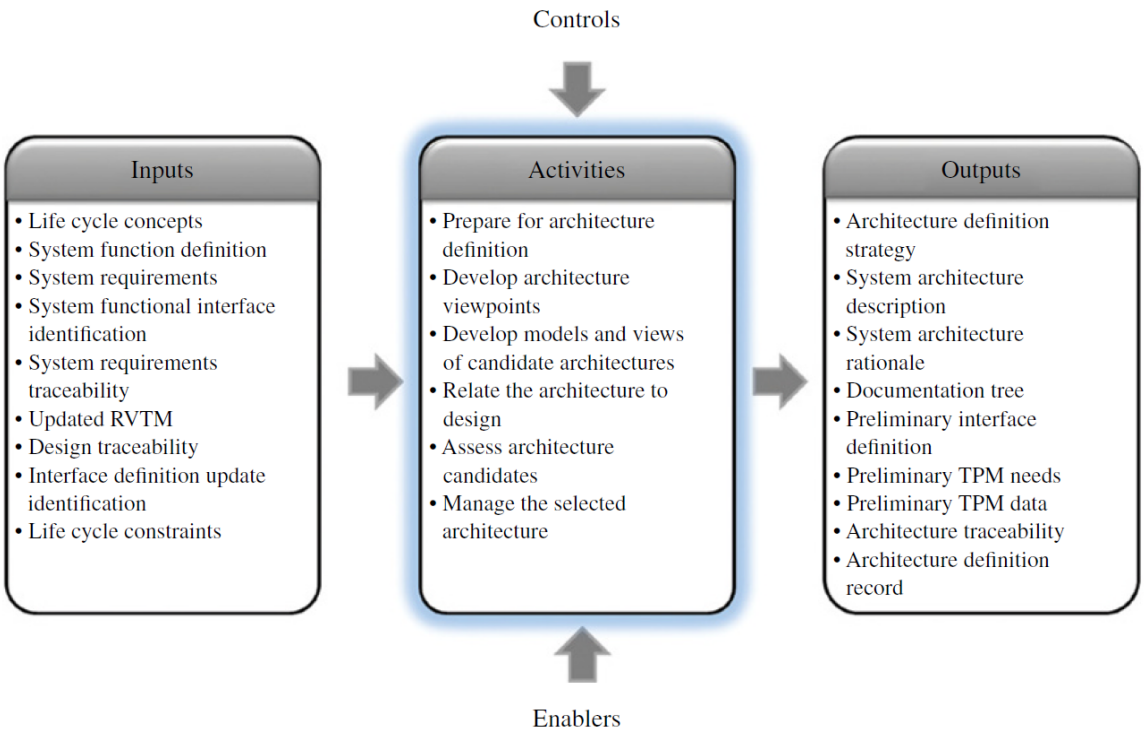

Figure 4. *INCOSE Systems Engineering Handbook* Architecture Definition Process. Source: INCOSE (2015, 65).

The *INCOSE Systems Engineering Handbook* (INCOSE 2015) also provides guidance for tailoring of the SE processes to apply them for specific use cases (145–149). An MBSE approach to projects should tailor the traditional systems engineering processes as MBSE methods enable new capabilities that were not available in a more traditional document-based approach. However, the tailoring needs to also remain focused on the SE principles the *INCOSE Systems Engineering Handbook* documents, but it should extend the methods defined for each SE activity. Figure 5 demonstrates INCOSE's method for tailoring the SE process with inputs, activities, and outputs (2015, 146). Developers can use the SE tailoring process from the *INCOSE Systems Engineering Handbook* for incorporating MBSE methodology and principles in their systems engineering life cycle processes.

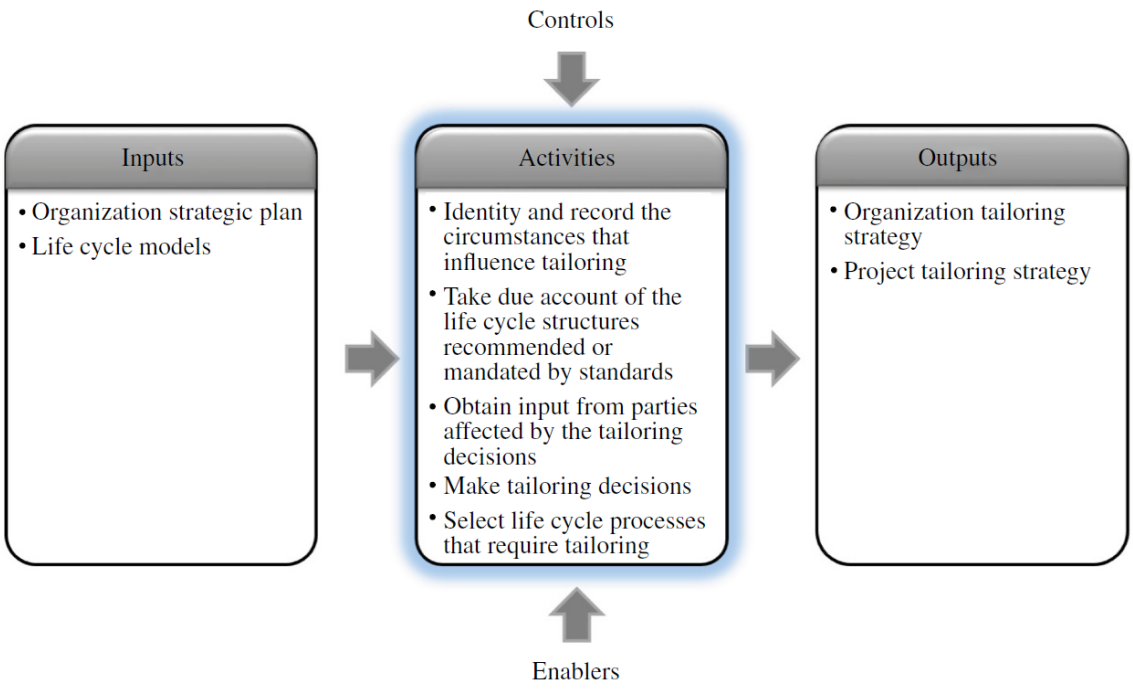

Figure 5. INCOSE Systems Engineering Process Tailoring. Source: INCOSE (2015, 146).

2. Defense Acquisition Guidebook

The DAG (DoD 2018a) provides guidance for the defense acquisition life cycle to compliment DoD Directive 5000.01 (DoD 2007) and DoD Instruction 5000.02 (DoD 2017). The guidebook does not give mandates but provides methods that program offices can tailor to meet the needs of individual programs. The DAG, as opposed to the *INCOSE Systems Engineering Handbook*, focuses on acquiring systems vs. developing systems. The DAG provides four main categories of guidance: acquisition methodologies; phases of the acquisition life cycle with activities, inputs, and outputs; technical reviews and audits; and systems engineering processes (DoD 2018a). The DAG gives insights into methods for systems engineering topics such as System of Systems Engineering, Software Engineering, Modular Open Systems, and Models and Simulation with specifics how to manage these aspects for an acquired system (DoD 2018a). An MBSE framework for an acquired system will need to address the concerns the DAG raises for systems engineering in addition to managing the acquisition of digital work products that describe or analyze the system.

The DAG provides details on several versions of acquisition models to address development needs over the acquisition life cycle for different types of systems (DoD 2018a). The DoD intends for these acquisition models to meet the needs of the stakeholders and the concerns from different types of systems. Stakeholders may require capabilities quickly from new systems or from updates to existing systems to meet emerging threats. Figure 6 shows the main version of the DoD life cycle that each of the others branch from (DoD 2018a). The DAG also presents alternative acquisition plans for software-intensive systems vs. hardware intensive systems (DoD 2018a).

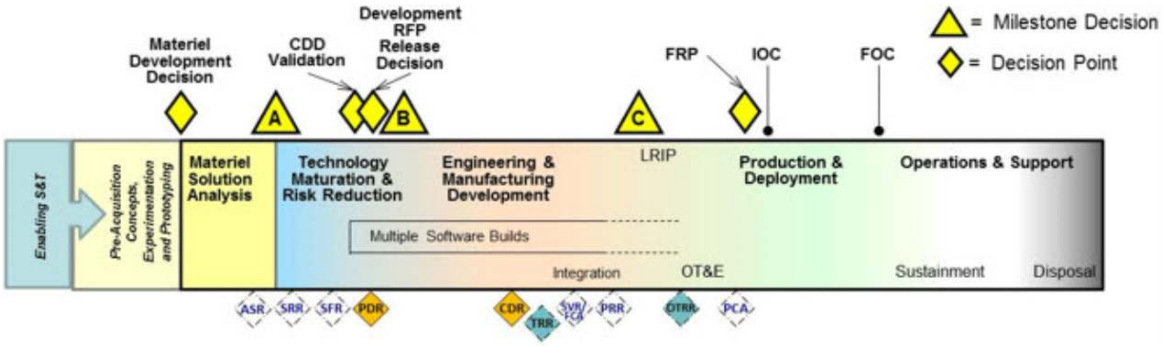

Figure 6. DoD Weapon System Development Life Cycle.
Source: DoD (2018a).

The Software Intensive Program model shown in Figure 7 represents some beneficial aspects for an MBSE approach. MBSE work products are similar to software products because they are both digital, malleable, and engineers can use them to evaluate the design early with portions of the final product being prototypes or abstract representations. The life cycle shows several builds occurring before an integration. With digital work products describing a system, system developers can create them early in the life cycle as compared with real hardware, with numerous iterations.

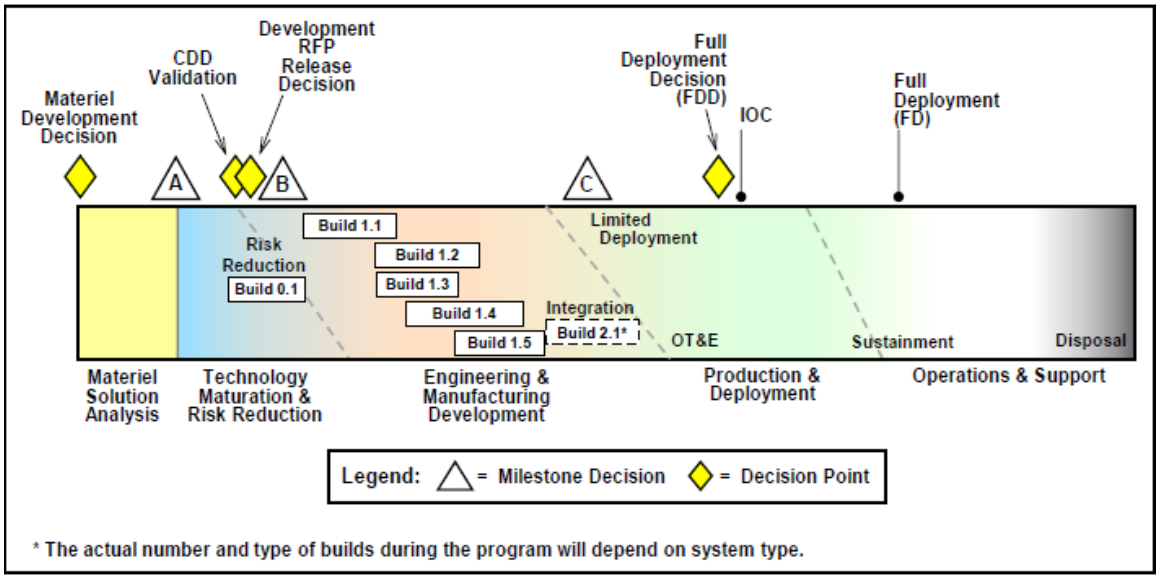

Figure 7. Defense Unique Software Intensive Program Acquisition Model. Source: DoD (2017).

The most important references from the DAG for an MBSE Framework are the technical reviews and milestone decisions. Figure 6 identifies the technical reviews along the bottom of the figure and the milestone decisions along the top. The DAG details what is important to review and the criteria to move forward on a program to reduce risk at each technical review (DoD 2018a). An MBSE Framework will need to accomplish the same goals of these technical reviews, even if it accomplishes those goals with different methods.

The DAG presents the SE processes slightly differently than the ISO 15288. As shown in Table 1, the DAG SE processes map to ISO 15288 SE processes (DoD 2018a). Unfortunately, The DAG does not include every 15288 SE process. The missing processes are from the agreement and organizational enabling processes Figure 3 identifies. Some of the process do not align perfectly such as interface management. With MBSE, architecture and interfaces are heavily entwined, and developers should manage them together.

Table 1. DAG SE Processes Mapped to ISO 15288 SE Processes.
Source: DoD (2018a).

	DoD SE Processes (DAG section)	ISO/IEC/IEEE 15288 Processes (Clause)
TECHNICAL MGMT PROCESSES	Technical Planning (4.1.1)	Project Planning (6.3.1)
	Decision Analysis (4.1.2)	Decision Management (6.3.3)
	Technical Assessment (4.1.3)	Project Assessment and Control (6.3.2)
		Measurement (6.3.7)
	Requirements Management (4.1.4)*	System Requirements Definition (6.4.3)
	Risk Management (4.1.5)	Risk Management (6.3.4)
	Configuration Management (4.1.6)	Configuration Management (6.3.5)
	Technical Data Management (4.1.7)	Information Management (6.3.6)
	Interface Management (4.1.8)	Architecture Definition (6.4.4)
	Covered by Section 4.3.18 PQM and DAG CH 1 Section 4.8 Encouraging a Quality Focus	Quality Assurance (6.3.8)
TECHNICAL PROCESSES	*Covered by Sections 2.3 System Level Considerations and 3.1.2. SoS*	Business or Mission Analysis (6.4.1)
	Stakeholder Requirements Definition (4.2.1)	Stakeholder Needs and Requirements Definition (6.4.2)
	Requirements Analysis (4.2.2)*	System Requirements Definition (6.4.3)
	Architecture Design (4.2.3)	Architecture Definition (6.4.4)
	Implementation (4.2.4)	Design Definition (6.4.5)
		Implementation (6.4.7)
	Covered throughout Sections 3.2 SE Activities in the Life-cycle and 4.3 Design Considerations	System Analysis (6.4.6)
	Integration (4.2.5)	Integration (6.4.8)
	Verification (4.2.6)	Verification (6.4.9)
	Validation (4.2.7)	Validation (6.4.11)
	Transition (4.2.8)	Transition (6.4.10)
	Covered by DAG CH 4 Life Cycle Sustainment	Operation (6.4.12)
	Covered by DAG CH 4 Life Cycle Sustainment	Maintenance (6.4.13)
	Covered by Section 4.3.7 Demilitarization and Disposal and DAG CH 4 Life Cycle Sustainment	Disposal (6.4.14)

*DAG Chapter 3 Requirements Management and Requirements Analysis processes are covered by ISO/IEC/IEEE 15288 – System Requirements Definition

A DAG technical management process that does not align well with ISO 15288 SE processes is Interface Management. The interface management process captures internal and external interfaces and their requirements, verifies interfaces comply with their requirements, and produces views of the interfaces (DoD 2018a). The mapping has interface management mapping to architecture definition, but it should map to at least system requirements management, architecture definition, design definition, integration, and verification processes. Interface management is a good example of a process that extends throughout the life cycle and matures through many layers of abstraction. An initial interface description layer could describe the information that needs to be exchanged, the next layer could describe the message format, another layer could describe the communication bus protocols used to pass the message, and a final layer would describe the physical connection that enables the communication bus. An MBSE Framework should be able to enhance interface management by having richer definitions of interfaces and earlier compliance verification that interfaces will meet their requirements.

B. MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE MODELS

Several models exist for an MBSE process. The MBSE process models reviewed for this thesis remain at a high level of abstraction similar to the traditional SE models covered in the previous section allowing these models to be reused by a wide range of projects. The focus of these MBSE process models have also been on the design of a system using models, and how the digital artifacts can enhance the analysis of designs. Estefan surveyed popular methodologies for implementing MBSE for INCOSE (2007). The important aspect of the report is that there are several methodologies available for use, and they have different advantages and disadvantages (Estefan 2007).

1. Object-Oriented Systems Engineering Method

INCOSE has adopted the Object-Oriented Systems Engineering Method (OOSEM), and an INCOSE working group is actively expanding the knowledge on the method (Estefan 2007). OOSEM relies on a recursive process as Figure 8 portrays (Friedenthal, Moore, and Steiner 2014, 418–420). If developers are using the OOSEM process at a System of System level, the activities are to specify and verify systems (Friedenthal, Moore, and Steiner 2014, 418–420). If developers use OOSEM within a system, specifications and verifications for subsystems are the outcomes (Friedenthal, Moore, and Steiner 2014, 418–420). The OOSEM phases align well with the phases of the DoD phases, but it is not a direct relationship with the milestone reviews. The conceptual design phase conclusion would better align with the DoD's System Functional Review (SFR), but the preliminary design and detailed design phases align with the conclusion of the Preliminary Design Review (PDR) and Critical Design Review (CDR), respectively.

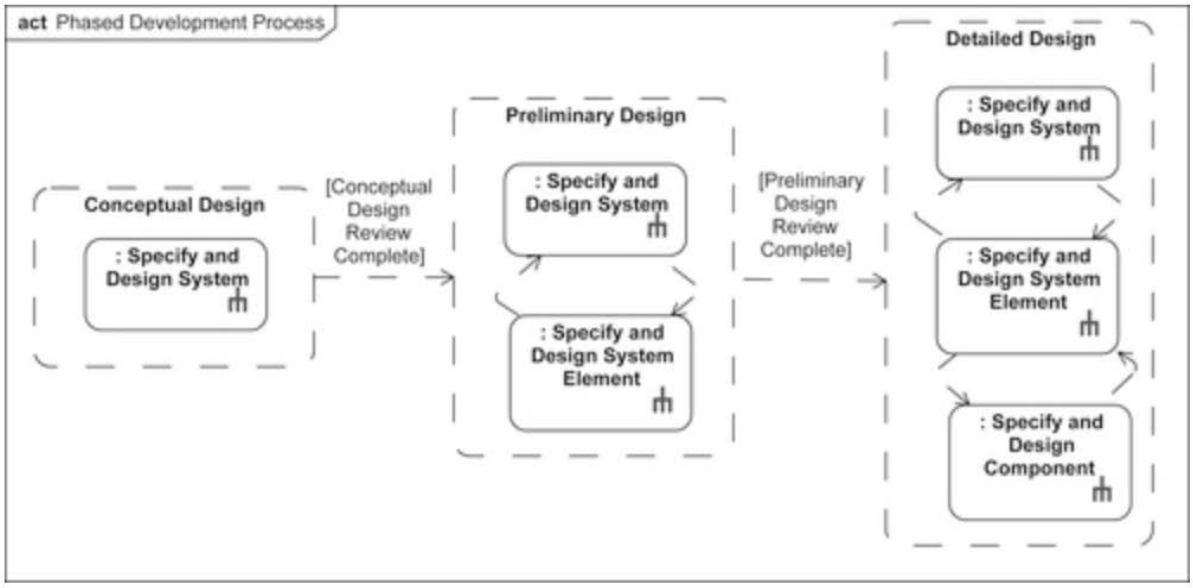

Figure 8. OOSEM Recursive Process. Source: Friedenthal, Moore, and Steiner (2014, 420).

DoD developments would need to specialize the OOSEM life cycle process. With many DoD systems, the DoD uses efforts to create advanced hardware prototypes in the earlier phases of the life cycle to reduce risk. System developers must account for these hardware prototypes because they are often difficult to change at a preliminary design abstraction level. The OOSEM phases also leave out the operational, sustainment, and disposal phases of the life cycle. System developers should use other methods outside of OOSEM to handle these later life cycle phases.

OOSEM starts with a traditional SE process at its top layer. Figure 9 identifies the activities and demonstrates their focus on architecture products. The method begins with the set-up model function which includes organizing the structure for the data of the architecture model and setting conventions for modeling (Friedenthal, Moore, and Steiner 2014, 423–431). Modeling conventions become extremely important with a digital descriptive model. Systems Engineers can automate verification and analysis with digital models if the model developers use a common metamodel or ontology throughout the models. When working from the aspect of a DoD acquisition program, this is important because a prime contractor or several prime contractors design much of the architecture.

The DoD will need to analyze a system design in the context of a domain of System of Systems, and integrate modeling data from several programs and/or prime contractors.

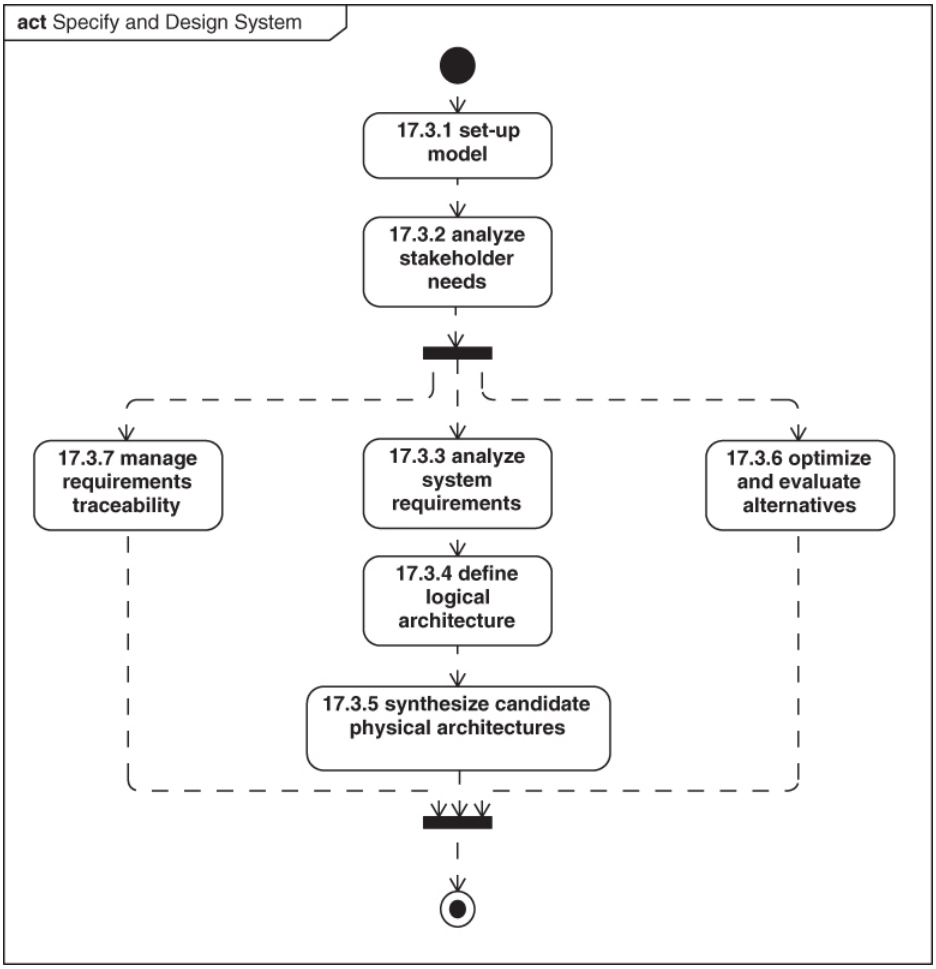

Figure 9. OOSEM Activities. Source: Friedenthal, Moore, and Steiner (2014, 423).

The method continues with the analyze stakeholder needs activity which includes functions to define the as-is domain, perform analysis, specify requirements, define use cases, and capture measures of effectiveness (Friedenthal, Moore, and Steiner 2014, 431–440). Defining the as-is domain involves identifying known systems in the domain and the interfaces between these systems and the desired capability (Friedenthal, Moore, and Steiner 2014, 433–436). The next activity is to perform analysis to identify the capabilities and important parameters for the system as a black box. This analysis leads to specifying

mission requirements (Friedenthal, Moore, and Steiner 2014, 436). The next activities can happen in parallel to define use cases, the key behaviors of the system as seen by the customer/user, refining the domain to the to-be domain, and capturing Measures of Effectiveness (MOE) (Friedenthal, Moore, and Steiner 2014, 432–440). The MOEs align with requirements/needs defined earlier but defines quantifiable parameters the developers can test or analyze the system against (Friedenthal, Moore, and Steiner 2014, 437).

The analyze system requirements activity defines the system requirements, but it also defines in what states for the system will be (Friedenthal, Moore, and Steiner 2014, 441–454). A key step in the activity is defining scenarios the system will be under as a black box system to identify the required functionality and interfaces for the system (Friedenthal, Moore, and Steiner 2014, 441–444). The systems engineers define scenarios using functional architecture products. A key aspect of OOSEM is that systems engineers develop a functional architecture prior to defining system requirements. The systems engineers then use the functional aspects of the system, along with design constraints, as inputs to define requirements (Friedenthal, Moore, and Steiner 2014, 447–449). Once the engineers define requirements, they define the states the system will be under during operation. Developers would then conduct a trade study to analyze how variations on the requirements might affect system performance (Friedenthal, Moore, and Steiner 2014, 453). Another ongoing requirement activity is for systems engineers to manage requirements traceability. With OOSEM, systems engineers trace requirements to not only parent requirements and test activities but also to the structural and functional architecture elements that refine or satisfy those requirements. Systems Engineers then conduct analysis on these relationships as well to identify gaps in the architecture or design functionality that goes beyond what the requirements state (Friedenthal, Moore, and Steiner 2014, 447–449).

The next step in OOSEM is to define a logical architecture (Friedenthal, Moore, and Steiner 2014, 454–460). A logical architecture allows system architects to analyze the architecture with structural components prior to defining how those components will be implemented. An example would be a sensing component that must identify objects on the battlefield. There are many ways that this could be implemented based on requirements

such as with a radar system, a global satellite system, or with a drone. The logical architecture allows an architect to analyze what the information and fidelity of information are required from the sensing system is and what interfaces would be required to other system components. The logical architecture also enables early interface analysis to be conducted to identify where complex interfaces may be located, or what type of interfacing structure is desired. Developers may weigh the benefits of a point-to-point interface between components versus a central communication system to which all components connect.

The final step in OOSEM is to synthesize candidate physical architectures (Friedenthal, Moore, and Steiner 2014, 460–488). Developers realize the logical architecture with physical components using defined criteria to maximize the effectiveness of the physical architecture (Friedenthal, Moore, and Steiner 2014, 460–463). Developers then refine the design of the physical architecture using software, data, and hardware design principles and toolsets (Friedenthal, Moore, and Steiner 2014, 478–484). Developers next define and review the specifications for the physical components (Friedenthal, Moore, and Steiner 2014, 484–486). A difficult aspect with the toolsets available today is managing the linkages between the information in the architectural models and the information captured in design packages.

While the activities to define requirements, logical architecture, and physical architecture are occurring, OOSEM defines another activity to perform analysis on the design as it is captured. In *A Practical Guide to SysML*, this is only defined as using parametric diagrams in SysML (Friedenthal, Moore, and Steiner 2014, 491–493). However, more complex analytical tools are truly required to capture this analysis.

2. "Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions" (Stepanchick 2016)

Lieutenant Commander (LCDR) Stepanchick's (2016) thesis researches MBSE methods and trade studies in the DoD acquisition life cycle. The thesis centers on adjusting the life cycle used by Strategic Systems Programs (SSP) to incorporate MBSE methods. In reviewing the application of the methods presented, LCDR Stepanchick refines the trade

space exploration process with examples of how the MBSE methods would be implemented to support the early life cycle activities to identify beneficial new architecture concepts (Stepanchick 2016).

Of interest for the work in this thesis, LCDR Stepanchick (2016) presents a different representation of the traditional systems engineering "Vee" as shown in Figure 10 Stepanchick's representation shows three different "Vees." Each "Vee" represents the traditional SE "Vee," but as the life cycle progresses, the "Vees" deepen to represent additional detail being incorporated into the system design. As with the traditional SE "Vee," these "Vees" should not be thought of as a strict waterfall approach, but iteration and recursion to further the progress along the "Vee." The result does a good job of identifying how MBSE can support early verification and validation of the system design as early as during the concept development phase of the life cycle. This concept mirrors methods from Agile Development and Software Engineering to build testable systems early in the life cycle. The virtual aspect of MBSE enables these principles before hardware is manufactured.

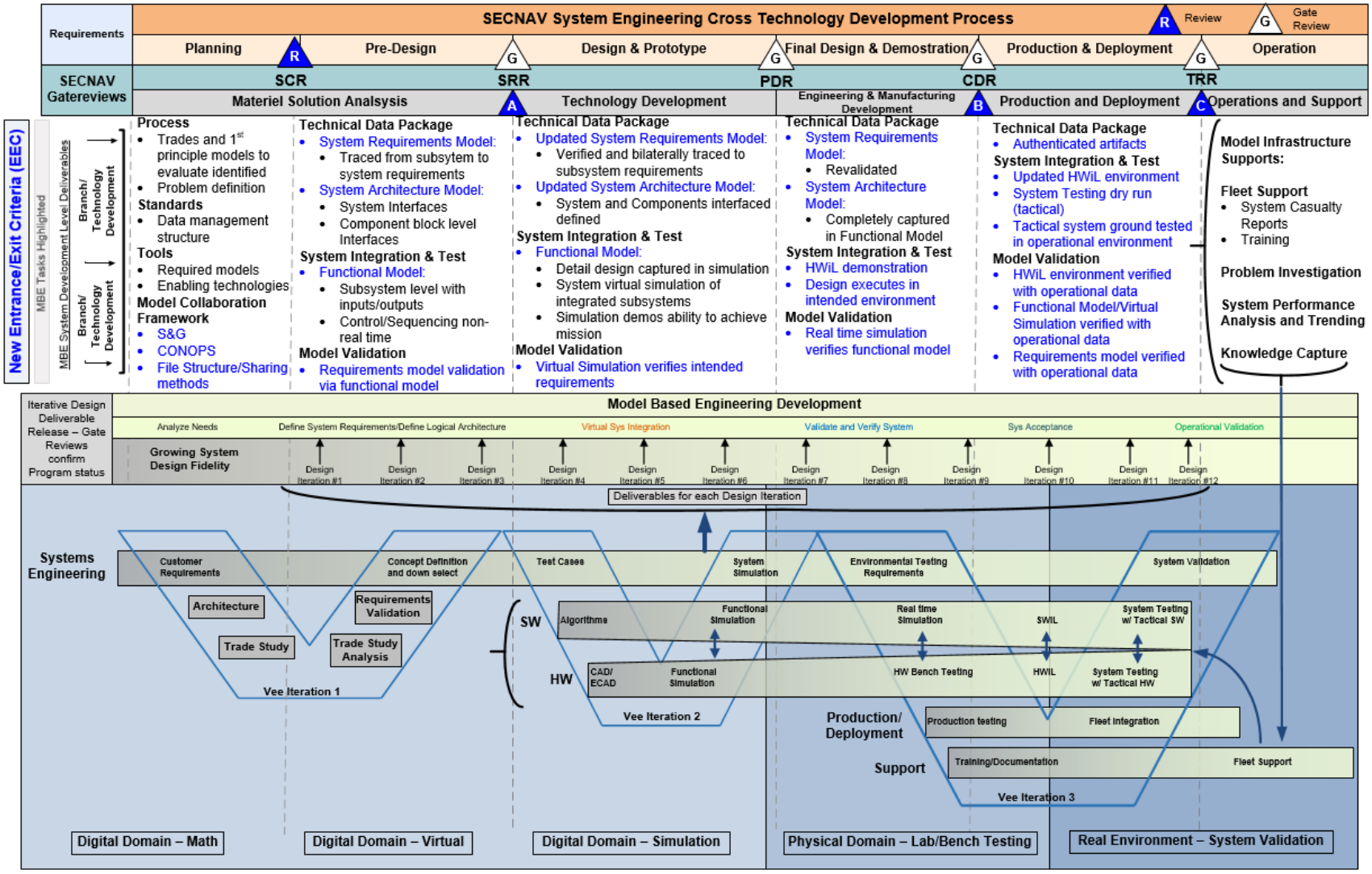

Figure 10. MBSE Integration into Acquisition Life cycle. Source: Stepanchick (2016).

Stepanchick (2016) presents the first “Vee” occurring during the activities prior to System Requirements Review (SRR). The first type of models created are trade space models. These models would provide objective quality evidence (OQE) that design decisions on the concepts developed to meet stakeholder needs are justified (Stepanchick 2016). The next model would be a low fidelity simulation of the concept. This model would be used to validate the requirements, identify infeasible requirements, and develop an understanding of requirement cost drivers (Stepanchick 2016).

Stepanchick (2016) presents the second “Vee” during the activities between the SRR and the PDR. During this phase, the system concepts are refined, the major subsystems are identified and allocated functionality, and the interfaces between these subsystems are defined. The left side of this “Vee” uses descriptive models to play a significant role during this phase to capture the system design. The right side of this “Vee” uses analytical models and simulations to test the design to ensure that it is meeting the requirements.

The final and third “Vee” Stepanchick (2016) presents is occurs between PDR and the operation of the system. The left side of the “Vee” develops capabilities to test the system. The method uses hardware in the loop (HWIL) and software in the loop (SWIL) to test portions of the system prior to full system production. The SWIL and HWIL capabilities allow developers to validate portions of the completely digital simulations created previously (Stepanchick 2016). The right side of this last “Vee” aligns with the right side of the traditional SE “Vee” to integrate and test the system after CDR and production begins. As the HWIL, SWIL, and integrated system testing improve and validate the digital functional simulations, these simulations can now be used to test the system under conditions, such as destructive testing, not affordable with real hardware.

Stepanchick’s (2016) SE process brings to the forefront many of the benefits of MBSE with multiple iterations of system design and continued verification and validation throughout the acquisition life cycle. Acquirers and developers wanting to implement an MBSE Framework can benefit from these concepts to gain the most benefits from MBSE methods.

III. MODELING A MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE PROCESSES

This chapter will cover a process to model an MBSE framework for managing the acquisition of a system. The next chapter will cover an example implementation. Figure 11 below shows a process for the life cycle definition team to define an MBSE framework. The process is a modification of the OOSEM process for developing a system from A Practical Guide to SysML (Friedenthal, Moore, and Steiner 2014, 417–503). The process shown is a linear process, but modelers should use iteration as they require.

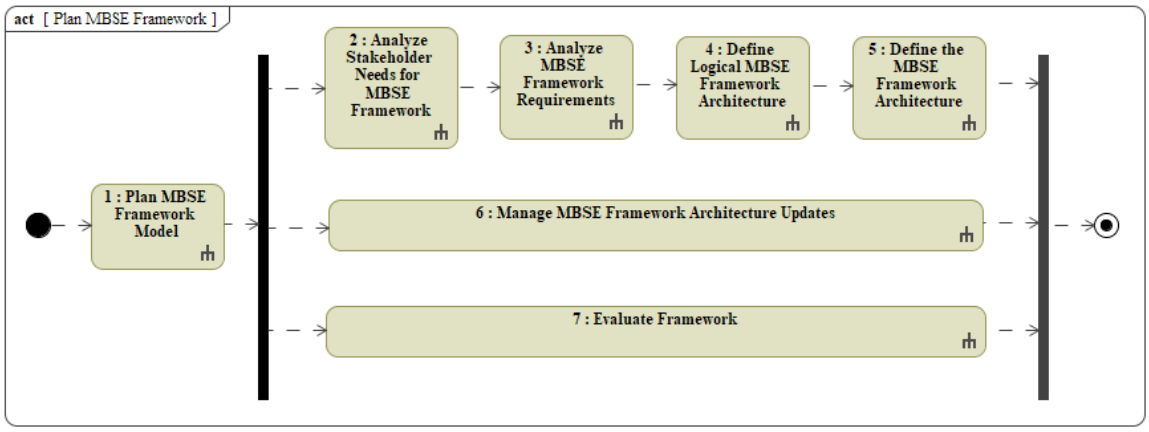

Figure 11. MBSE Framework Development Process

An architecture model can aid an organization in similar ways as a system architecture model would. The model can trace decisions in the design of the life cycle model to requirements, rationale, or stakeholder needs to understand the importance of acquisition functions. Interrelationships between items, structure, and behavior can trace in multiple dimensions to allow users of the process to view different aspects based on their viewpoint of the process. A test engineer has different needs from the process than a logistics engineer, and a model can assist in creating these alternate views while maintaining integration between the views.

One benefit of modeling the life cycle is the ability to link to previous standards or life cycles created by the organization. An organization can treat the processes described in Chapter II as abstract processes that their specialized acquisition life cycle can trace to them. Users of the created life cycle process can see the content from the referenced life cycles to identify additional concerns, methods, and potential products they should create when executing a systems engineering activity. Figure 12 shows an example mapping of a portion of the DAG systems engineering processes to the OOSEM process provided in A Practical Guide to SysML (Friedenthal, Moore, and Steiner 2014, 417–503). An organization can use this process to specialize their life cycle model for a project with an upfront plan for how systems engineering activities will occur.

Modeling the life cycle has a potential pitfall as with modeling a system: the life cycle model needs a clear purpose and scope to ensure the modeling does not become frivolous. A modeler may want to over document the activities that need to occur over the life cycle but must remember that one aspect of moving to MBSE is to remain agile. The life cycle model must have enough flexibility to enable program managers to diverge from the documented life cycle. The examples in Chapter II provide good guidance to the level of detail needed for an organization's model. A project model should have additional detail by allocating activities to organizations, providing the work products expected from the activities, and providing processes for procedures. Program management will need to add the level of detail they require in a tool familiar to the program management team.

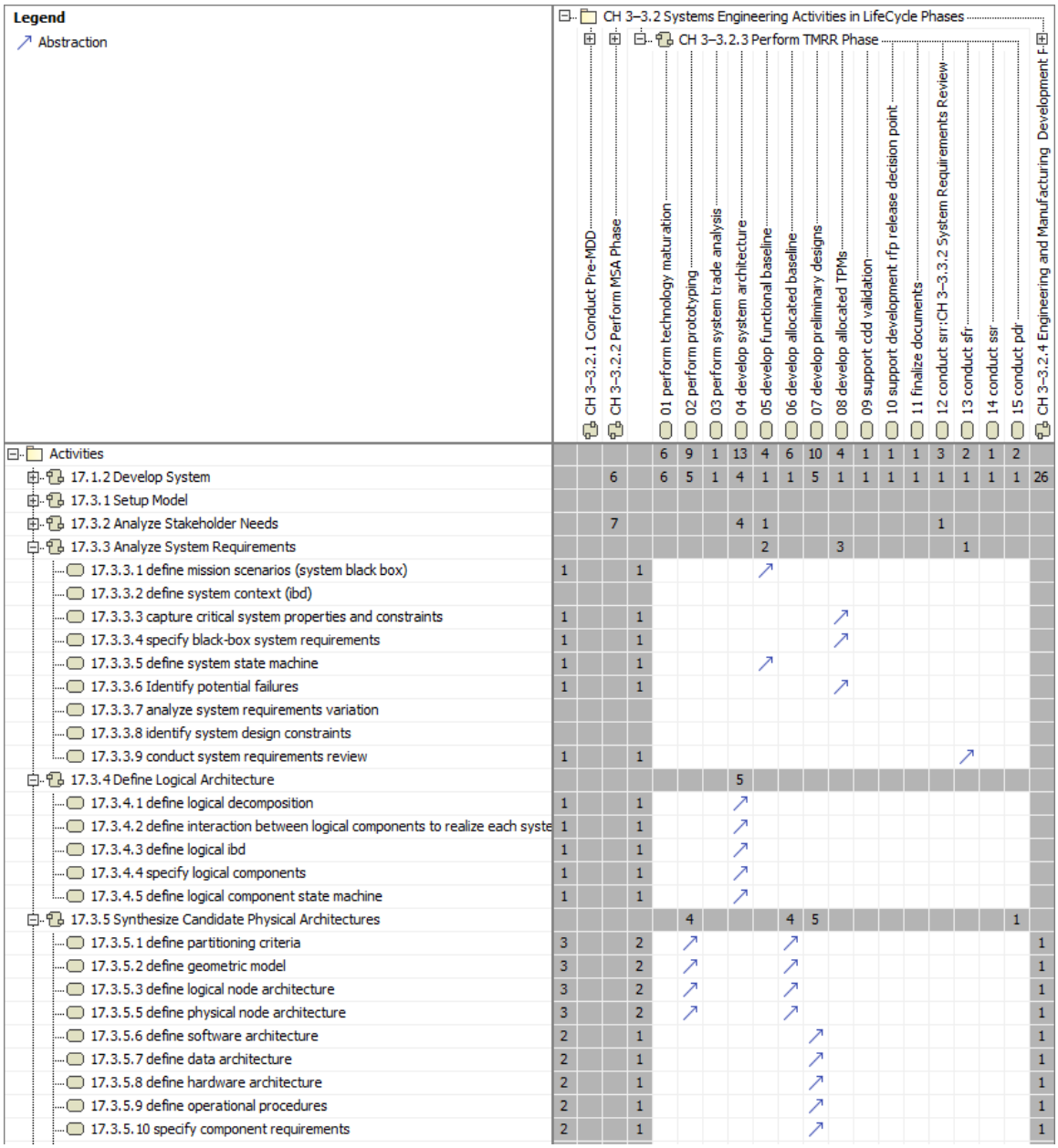

Figure 12. DAG Systems Engineering Process to OOSEM Process

A. MBSE FRAMEWORK MODEL PLAN

The first activity is the same as OOSEM and involves establishing modeling conventions and organizing the model. These conventions and organization can take many forms. INCOSE used a traditional document format, while the DAG uses an HTML web format with URL links between sections. This thesis uses a SysML model with a package structure and standardized element usage. A project can use any of the above formats that best suit the project's needs, but the life cycle model management should first establish the method for documenting the life cycle model.

For SysML, a modeler can use the package structure to simplify model navigation and to imply properties to the elements through containment. An example is with structural elements in for an acquisition process: One package can contain all the human elements of the development team, another contains the applications the development team will use, and a third can contain the data that the development team creates across the life cycle. Figure 13 depicts an example package structure for modeling a system (Friedenthal, Moore, and Steiner 2014, 430). The structure should be modified to meet the needs of the project. A consideration should be to create a package structure that is relatable to existing processes. As opposed to a modeling term to name packages such as "2-Structure" in Figure 13, the package names should have more relatable terms such as infrastructure and organizations.

Modeling conventions affect how a team will model the processes. SysML is a language, and modelers can use different elements and relationships to represent the same thought. In addition, the SysML standard enables modelers to extend the language as they require. When capturing conventions, SysML requirement elements should represent the guidelines as they are requirements for how the model is built. Table 2 shows a subset of potential modeling guidelines. If an organization is new to modeling and does not already have a robust set of architecting guidelines, they can start with a limited set. For undecided modeling methodology, the definition team can capture these guidelines as the modeling of elements and relationships first occur to ensure future modeling is consistent.

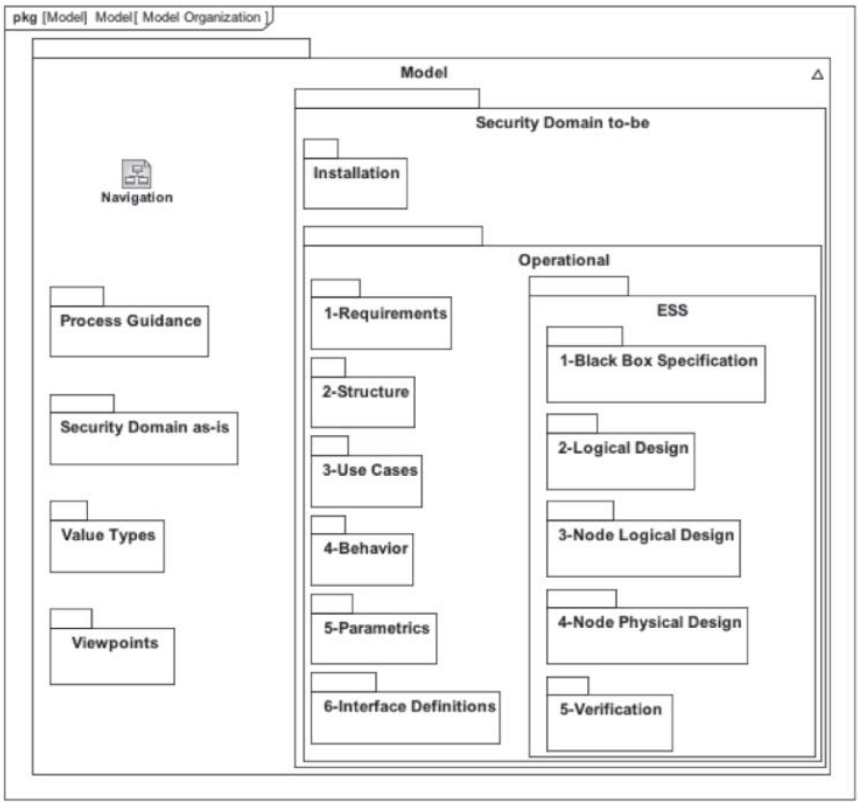

Figure 13. Example Package Structure.
Source: Friedenthal, Moore, and Steiner (2014, 430).

Table 2. Example Process Modeling Guidelines

ID	Guideline	Rationale
GUI-34	Names for blocks, activities, states, packages, diagrams, and value types shall use title case notation.	Title case distinguishes elements of classification from elements of usage.
GUI-43	Names for elements shall be alphanumeric with spaces.	Enables integration with other modeling tools for simulation or analysis.
GUI- 37	All actions shall have a behavior of type Activity.	
GUI-38	Pins on actions shall be typed by a block.	Pins that exchange information between actions should not be left untyped.
GUI-42	Activities shall either be owned by a use case or be used to type the behavior of an action.	An activity that does not use these criteria is not be used by the model and should, therefore, be deleted.

B. MBSE FRAMEWORK STAKEHOLDER NEEDS AND PROCESS REQUIREMENTS

Figure 14 depicts the activities to analyze stakeholder needs. The most important stakeholders will be the program office that is managing the acquisition and the contractor developing the system. The team should attempt to translate MBSE related stakeholder desires into needs as opposed to solutions. An example is a stakeholder requesting a HWIL solution instead of requesting validation critical systems will meet requirements prior to full system integration.

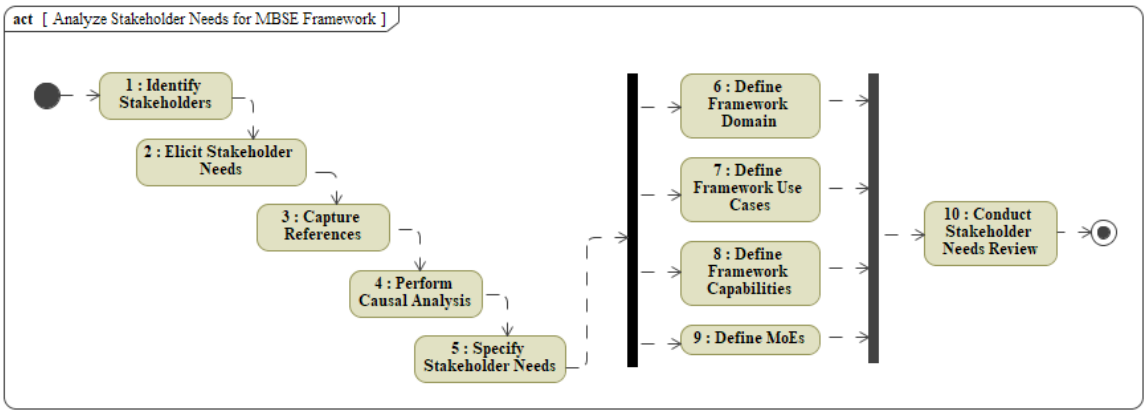

Figure 14. Analyze Stakeholder Needs for MBSE Framework

The needs of the prime contractor will be difficult to capture early in the life cycle because the program office will likely not have selected a prime by the time that they need to begin capturing their SE processes. The model will need to capture more generic contractor needs early and review these assumptions once the program office selects a contractor.

The activity “Capture References” should identify the MBSE and SE processes the MBSE Framework will rely for guidance. The references in Chapter II are a good start in addition to current program office process and standards for specific SE processes. The activity “Perform Causal Analysis” refers to analyzing these references for their capabilities and limitations (Friedenthal, Moore, and Steiner 2014, 434–436). The limitations are where MBSE methods may be able to aid. As discussed in Chapter I, Vice

Admiral Grosklags (2017) pointed to the delays in delivering capabilities to the warfighter as a limitation of the current SE processes the Navy uses.

The stakeholder needs are then formalized to drive decision-making. The stakeholder needs will be a set of SysML requirement elements, and they will guide the decisions made in the next steps. These stakeholder needs will be used in review to validate the modeled process is meeting its purpose.

The next steps are in parallel because they require iteration to complete. The first of these activities is to define the MBSE framework domain. The domain should include development team as a singular entity and the other parties that they will interact with over the course of the life cycle. The development team will consist of the program office, contractors, and other government national team members. Examples of other interacting parties would be the warfighter, milestone decision authority, test ranges, Weapon System Explosives Safety Review Board, and existing facilities that will interact with the system. An MBSE framework may include existing network and application infrastructure, such as the Naval Systems Engineering Resource Center (NSERC) or Integrated Modeling Environment (IME) if the program would not set up its own network.

Figure 15 provides an example for SysML use cases and the relationship to external actors using a SysML use case diagram. Most use cases will not change between a traditional acquisition process with use cases such as define system requirements, acquire the system, manage system specifications, test the system, and operate the system. MBSE functions should be explored for those that are desirable such as having a single repository for requirements, creating a digital twin, or developing an HWIL environment for the CDR with a capability element. SysML does not have an element for capabilities, but the language can be extended with a capability stereotype as a specialization of a SysML stereotype Block. These capabilities should have SysML association relationships between them and the use cases as Figure 16 displays.

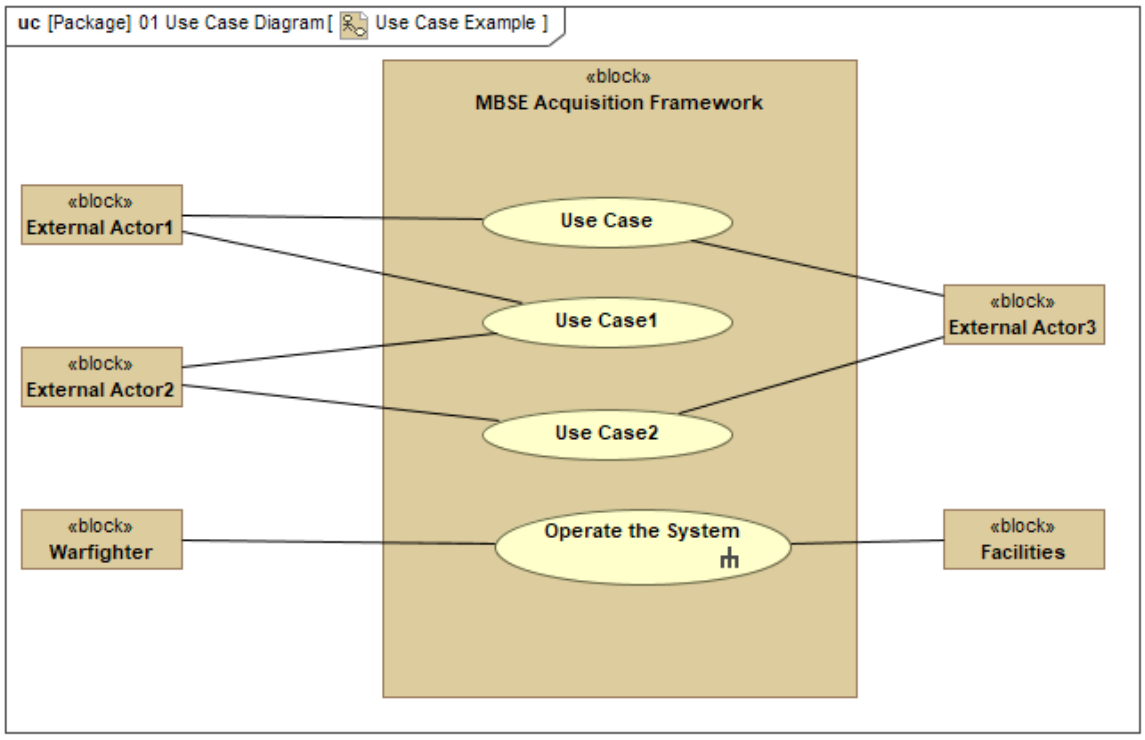

Figure 15. Example Framework Use Case Diagram

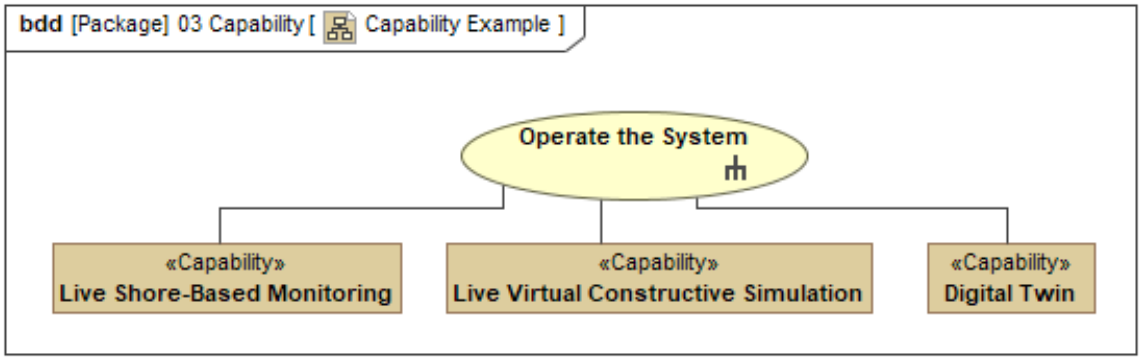

Figure 16. Relating Capabilities to Use Cases Example

Finally, an MoE for each stakeholder need should be defined. The MoEs can be defined with metrics concerning schedule, successfully meeting milestones, or the successful ability for a project to create desired capabilities using the MBSE framework. SysML does not have an element that aligns with an MoE so the requirement stereotype should be extended for a stakeholder need with the additional tag for an MoE.

At this point, the team with the relevant stakeholders should review the products created to ensure that leadership and the definition team are in alignment to create the desired MBSE framework capabilities and how they expect system developers to use those capabilities.

C. MBSE FRAMEWORK REQUIREMENT CAPTURE

Figure 17 depicts the activities to analyze MBSE framework requirements. In this step, scenarios are created for executing the work of the use cases from the previous activity to aid in defining the process requirements in addition to capturing any other constraints that apply to the process. The outcome of this activity will be a set of SysML requirements that will guide the rest of the life cycle definition process.

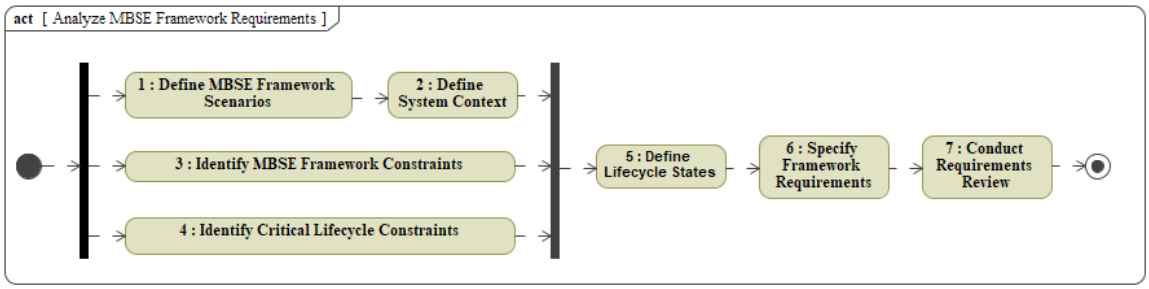

Figure 17. Analyze MBSE Framework Requirements

The first activities for analyzing requirements are in parallel because they each set the boundaries for the life cycle. The easiest of these is to define both the MBSE framework constraints and the critical life cycle constraints. For a DoD program, life cycle constraints will come from the DoDI 5000 instructions for what the Office of the Under Secretary of Defense for Acquisition & Sustainment requires of programs and from other federal or DoD regulatory documents. Another example would be the program needing to meet any applicable Security Classification Guide(s) (SCG) requirements. MBSE framework constraints will come from what MBSE capability is achievable. Examples of MBSE constraints may come from networks that will be usable to the development team due to classification constraints, available enterprise toolsets, existing system or environment

models that developers need to integrate into the MBSE framework, or enterprise modeling guidelines.

The next step to do in parallel is to define scenarios for MBSE framework. These scenarios will detail how the elements defined in the MBSE framework domain will interact, and they should take the form of activity diagrams in SysML. Figure 18 displays an example of the containment relationships between the use cases and the scenarios that type actions for the use case. The scenarios will mostly be from traditional SE processes. The level of detail for these scenarios should be similar to functions the references in Chapter II define at the highest levels such as define requirements, verify system, etc. The major MBSE impact here will be what format exchanged data is in if the external actors will take model-based information or require traditional reports. A program could fully define the information in a Systems Engineering Plan (SEP) in a set of models, but the Milestone Decision Authority requires a document format. Differences between native model data, intelligent exports, or document exports will have an impact on the MBSE framework functions.

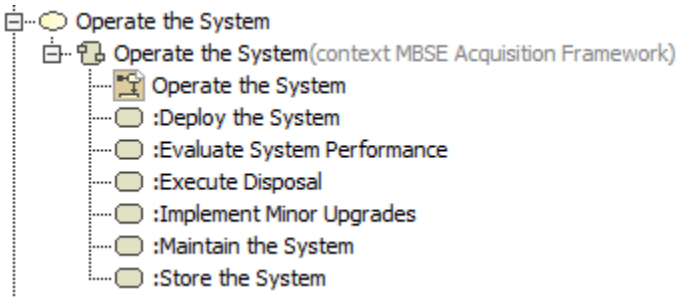

Figure 18. MBSE Framework Use Cases to Scenario Relationships

The next step is to define the acquisition life cycle states. Figure 19 shows a simple example for the states of the life cycle and the conditions to move between states. If states require further detail, the states can be modified, expanded or decomposed with submachine states. States can be used in future activities to define when functions occur.

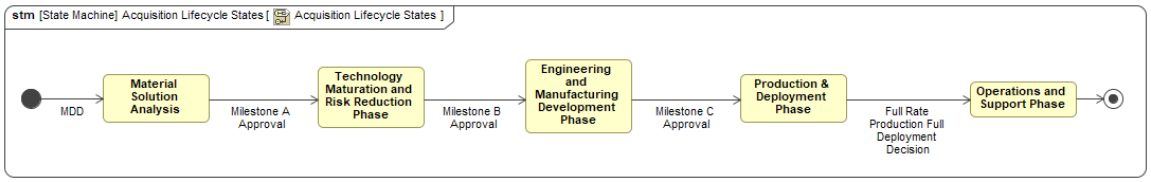

Figure 19. Example State Diagram

The requirements are defined for the process that will be a combination of the previously defined constraints and the requirements derived from the created scenarios. The requirements will be a combination of performance requirements of the life cycle functions and the information that must be produced. Finally, the requirements and the associated information should be reviewed prior to moving forward to the logical architecture.

D. MBSE FRAMEWORK LOGICAL ARCHITECTURE

Figure 20 depicts the activities to define the logical MBSE framework architecture. A logical abstraction of the life cycle process allows the modeling effort to consider the framework without the bounds of a real structure such as a program office or the tools the developers will use. The outcome of this activity is a WBS, tool needs, functions to complete a project, and work products the development team will create.

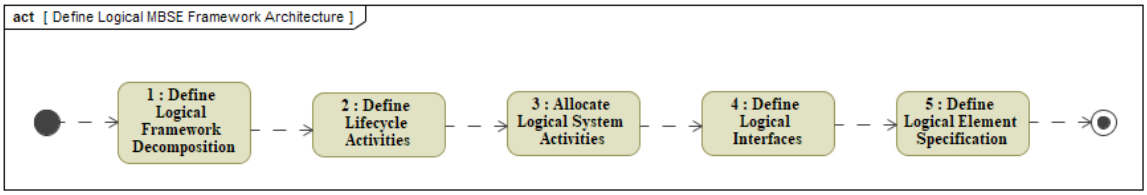

Figure 20. Define Logical MBSE Framework Architecture

The first step is to define the logical framework. The logical framework will be a combination of a work breakdown structure (WBS) and the infrastructure and tools the development team will require. The WBS structure should be similar to a traditional acquisition effort. The process may include specific MBSE roles to the WBS for oversite to ensure the program is adopting MBSE methods. Depending on the size of the program,

the MBSE role may be more beneficial at an organization level as opposed to the program level. The tools defined at the logical level should be by the capabilities the program will require such as a requirement management tool, a simulation tool, and a database for test and analysis results.

The next step is to decompose the activities from the scenarios into further refined activities. The model should decompose activities such as manage requirements into the activities to define, store, change, and analyze requirements. The interfaces between functions can then be defined. This is where adding full detail can become burdensome, as defining the entire data set and interactions for a program can be difficult. The level of detail for these activities should be similar to those in the references from Chapter II and other MBSE or SE processes. Finally, activities can be associated with the life cycle states by applying them as "do" activities of each state. The tables can be built to query model data to extract relevant information about functions.

With a logical structure and behavior architecture, behavior can be allocated to the WBS and information elements can be allocated to logical tools. Table 3 shows an example extracted from a model with functions to verify requirements at different levels.

Table 3. Example Logical Function Information

Action	Allocation	Activity	Activity Documentation	Input	Output
Verify system requirements	System Requirement Management	Verify Requirements	This activity verifies incoming requirements against the program's requirement modeling conventions. This step will also produce requirement metrics on the deriveReqt, satisfy, refine, and verify relationships and the number of requirements.	System Requirements Document	Requirement Verification Report
Verify launcher requirements	Launcher Requirement Manager	Verify Requirements	This activity verifies incoming requirements against the program's requirement modeling conventions. This step will also produce requirement metrics on the deriveReqt, satisfy, refine, and verify relationships and the number of requirements.	Launcher Requirement Document	Requirement Verification Report

The high-level interfaces between WBS activities can then be defined. The model can use multiple SysML internal block diagrams (ibds) to define these views, or query them from the functional actions. At the least, the information passed needs to be defined with SysML Blocks. If ibds are created, the SysML object flows between actions allocated to different sections of the WBS should align with item flows on connectors between WBS part elements.

E. MBSE FRAMEWORK PROCESS

Figure 21 depicts the activities to define an implementation for an MBSE framework architecture. A program office would use the result of this activity to define an organizational baseline process or to define a process for a specific project. The products would be a program office and contractor structure, infrastructure architecture, and plans and procedures to implement a project.

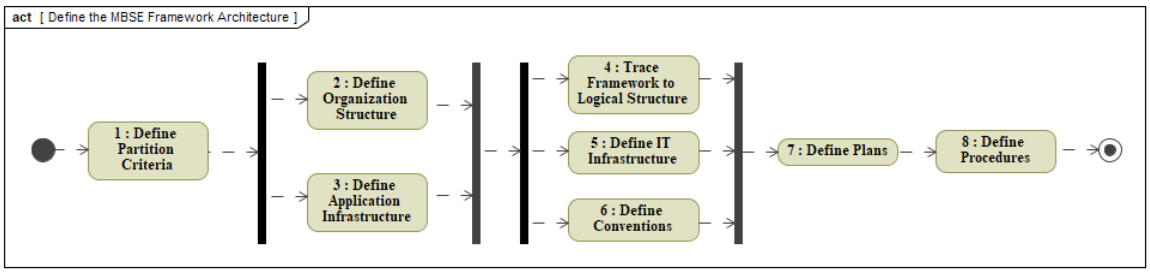

Figure 21. Define the MBSE Framework Architecture

The first step is to define partition criteria for allocating the logical WBS and tools defined in the logical MBSE framework architecture. Many of these criteria are similar for how program offices already allocate work to branches of the program office or between government and contractor responsibilities. Examples for these criteria would be to aggregate capability skillsets, reducing software application interfaces, or reducing program office branches that interface with external actors.

The next step is to define the organization and application infrastructure using the partition criteria. The organization structure will be the program office and any program office support, external government organizations, and the prime contractor. The

application infrastructure will be the specific applications the program office will use to model the system under development, configuration manage system information, and distribute information.

The organization and application infrastructure should then be mapped to the logical framework structure. This can be completed without creating a new relationship element between the implementation and logical elements if the original behaviors are reused. SysML actions are allocated to both WBS and Implementation elements, but the allocation of logical WBS elements to the organizations or infrastructure that will fulfill them are identified through model queries as opposed to a new relationship as Figure 22 shows with the chain relationship identifiers between Framework WBS Element and Framework Implementation Element. This automated relationship will be used in Chapter IV for automated views and analysis of the model.

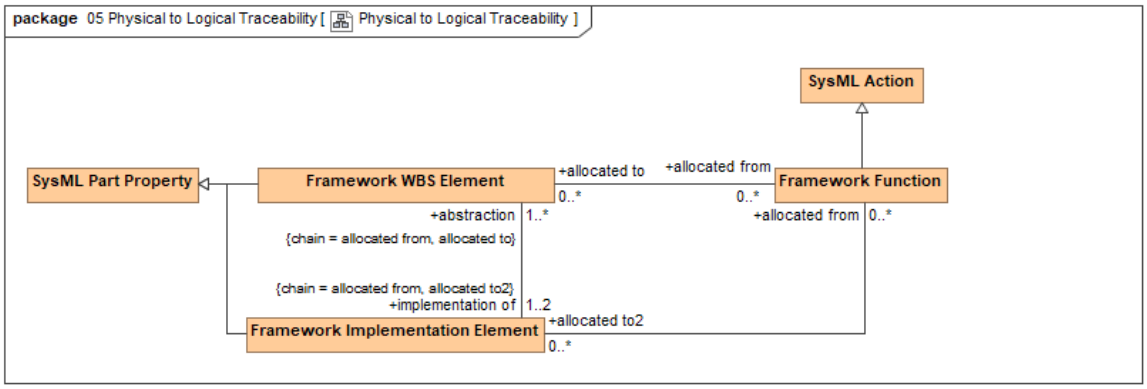

Figure 22. Implementation to Work Breakdown Structure Allocation Metamodel

The information technology (IT) infrastructure the program will use to enable collaboration can then be defined. This is a key step because the ability to quickly share information is a key tenet for MBSE. An infrastructure will need to allow the different organizations on the project to be able to exchange information quickly while meeting any SCG restrictions.

Once the applications infrastructure is known, conventions for models are developed for the models the development team creates. An organization can create general conventions, but the team must also set specific conventions for specific applications. For example, a general convention is to require traceability from derived requirements to parent requirements. However, for the application DOORS, this would require multiple derived conventions for how to set up DOORS link modules, for the hierarchy of requirement modules, and for attributes to display the traceability. However, for Cameo's SysML tool, the team would create a different set of derived conventions for a project's package structure, teamwork cloud project structure, how to use containment or the SysML derive relationship to show traceability, and how to use tables or matrixes to display requirement traceability. This would extend to the other type of MBSE applications the program would want to use such as analytical tools, simulators, HWIL, or SWIL.

The IT infrastructure for the program should then be developed, creating views for plans, and creating more detailed activities for procedures. Defining the IT infrastructure allows the process to define how team members will be able to collaborate on MBSE digital models. Program plans can leverage the architecture to provide a singular vision for how to develop the system. For procedures such as configuration management, the team can use the existing structure elements in the model for allocating responsibility and the tools in the procedures. The team should use an abstraction relationship between the highest-level functions for the procedures and the logical functions created during the define logical MBSE framework activity. An example would be creating relationships between a logical function to manage system specification configuration with procedures to manage system specification change, audit system information configuration, and maintain system specification baseline.

F. EVALUATE AND MAINTAIN MBSE FRAMEWORK MODEL

The last two activities are to manage the MBSE framework updates and to evaluate the MBSE framework. Organizations should update the framework model both during a development cycle and between projects. DoD Instruction 5000.02 requires programs to provide a SEP at Milestones A, B, and C for approval (DoD 2017). The program should

review the framework model for updates in conjunction with the SEP updates. Upon completion of projects, organizations should evaluate the development team's performance and lessons learned to help identify processes in the MBSE framework to target for improvements.

Organizations should evaluate the framework model-based on how the model meets the guidelines created during plan the framework model activity and for how well projects perform as compared to the MoEs captured during the analyze stakeholder needs activity. The team should be able to use the guidelines to find deficiencies in the model with respect to traceability and to find unused elements of the model similar to efforts to eliminate dead code in software. Organizations should also be reviewing how new methods and tools that are emerging from MBSE could benefit their efforts.

IV. MODELING A MODEL-BASED SYSTEMS ENGINEERING LIFE CYCLE PROCESS

This chapter explores the use of the process defined in Chapter III to define a program and the infrastructure that would be set up for a MBSE Framework. The project modeled a middle tier of acquisition (MTA) (Section 804) program to allow flexibility in defining the systems engineering process to be more MBSE focused. "The MTA pathway is intended to fill a gap in the [Defense Acquisition System] for those capabilities that have a level of maturity to allow them to be rapidly prototyped within an acquisition program or fielded, within five years of MTA program start" (DoD 2019). An MTA authority relieves a program of needing to comply with more complex requirements from a DoDI 5000.02 (DoD 2017). Figure 23 shows the differences in phases for a MTA program vs a traditional program. DoDI 5000.80 simplifies the phases to a rapid prototyping and rapid fielding phases that can either move directly into sustainment or onto a traditional major capability acquisition (MCA) (DoD 2019). The lack of definition for an MTA's systems engineering process also gives this work applicability to be used for an MTA program that needs to define how it will operate given the additional freedom an MTA can have. The model is not a complete model but shows examples of the full content that should be created. The model can be used as reference and expanded upon for use to model a program to a higher level of fidelity.

The system chosen to be developed under the MTA was a missile system that would be integrated on currently available platforms. The missile system is set as a mature technology, but still requires rapid prototyping to become a fieldable system. Some technologies inside the missile will be updated for manufacturing or performance reasons, and other auxiliary systems such as weapon control and logistics systems need to be developed from initial concepts. This allows the process to define a system that has elements that are at different phases of an MCA. This impacts how a program would track development of a system with different expectations of maturity levels at Systems Engineering Technical Reviews (SETR).

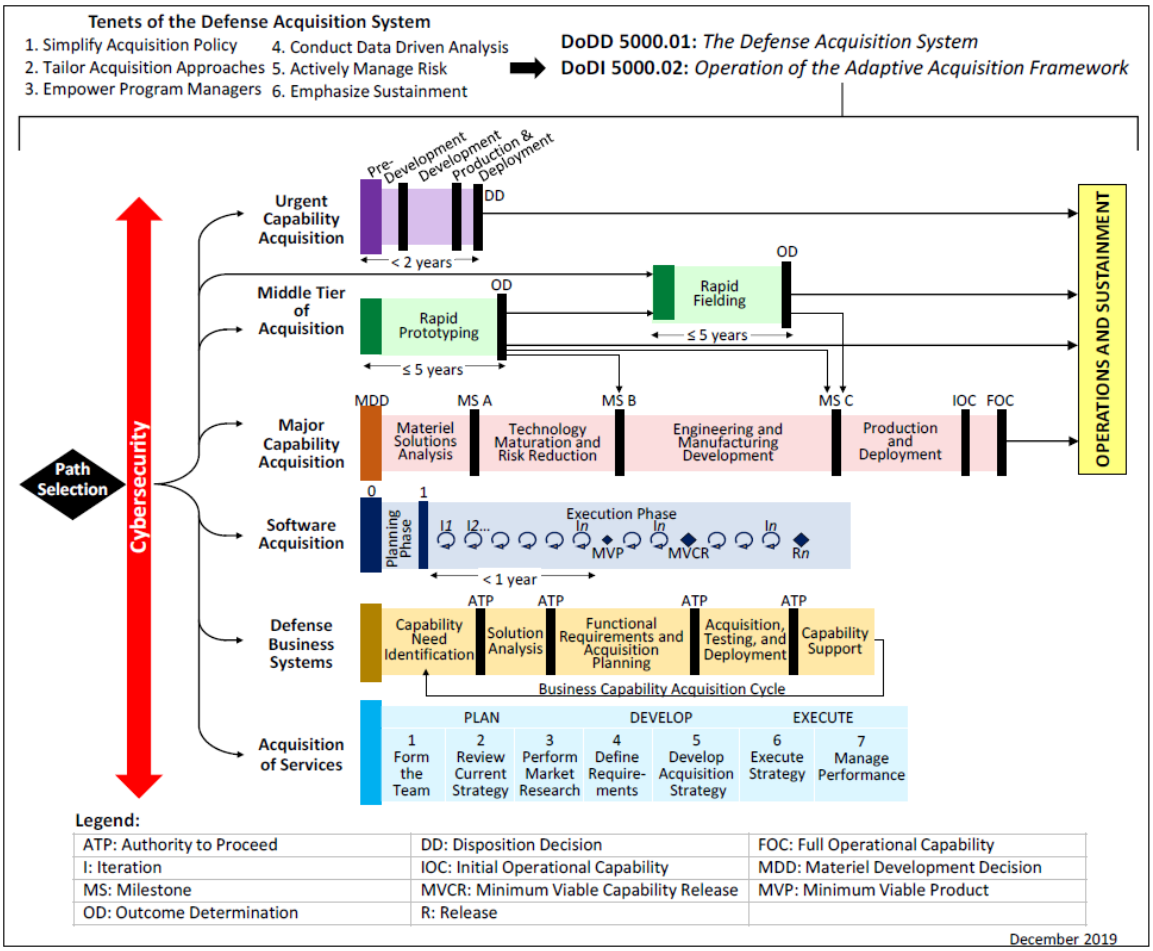

Figure 23. Adaptive Acquisition Framework. Source: DoD (2019).

A. MBSE FRAMEWORK MODEL PLAN

The project followed the process by first planning out the architecture model. The main two artifacts of the planning activity are a set of guidelines for the model to adhere and a package structure for organizing the model content. The project defined initial guidelines and package structure according to best practice, but as the model grew, the modelers added additional guidelines and modified the package structure. Additional guidelines are easy to add. However, guideline or package structure changes require may require rework. The project more than doubled the number of guidelines and clarified guidelines that were ambiguous but made few fundamental changes to existing guidelines or package structure. Because many of the guidelines are essentially another layer of the underlying metamodel built on top of SysML, modelers should expect to evolve those

aspects of the metamodel based on the needs of the project. This is especially true when trying to model new types of models such as this programmatic model.

Because the project decided to use SysML, guidelines were broken into the diagram views SysML has available, additional views used by the chosen tool, No Magic's Cameo System Modeler, and general guidelines used across the diagram views. Figure 24 shows the overview of the types of guidelines created. The guideline structure grouping aided modelers as they were building the model through the views SysML allows.

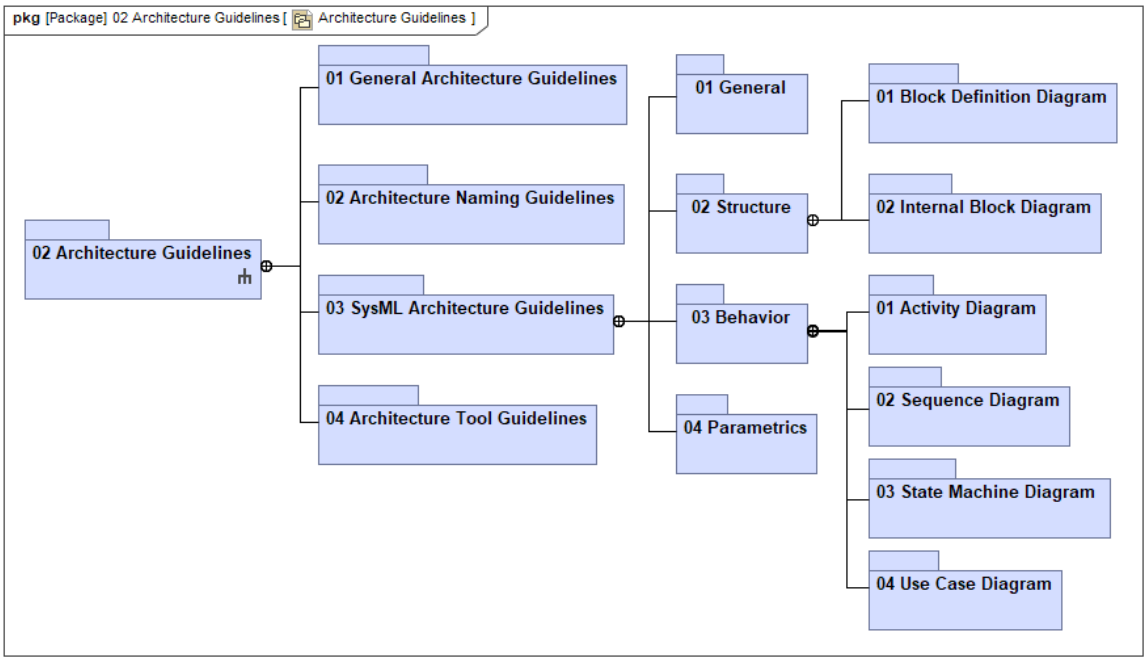

Figure 24. Modeling Guidelines Structure

The project based the package structure on the example from *A Practical Guide to SysML* (Friedenthal, Moore, and Steiner 2014, 430) as seen in Section III.A. This structure allowed the modelers to understand what sections of the model were being modified based on the process. Figure 25 visualizes the package structure from the model.

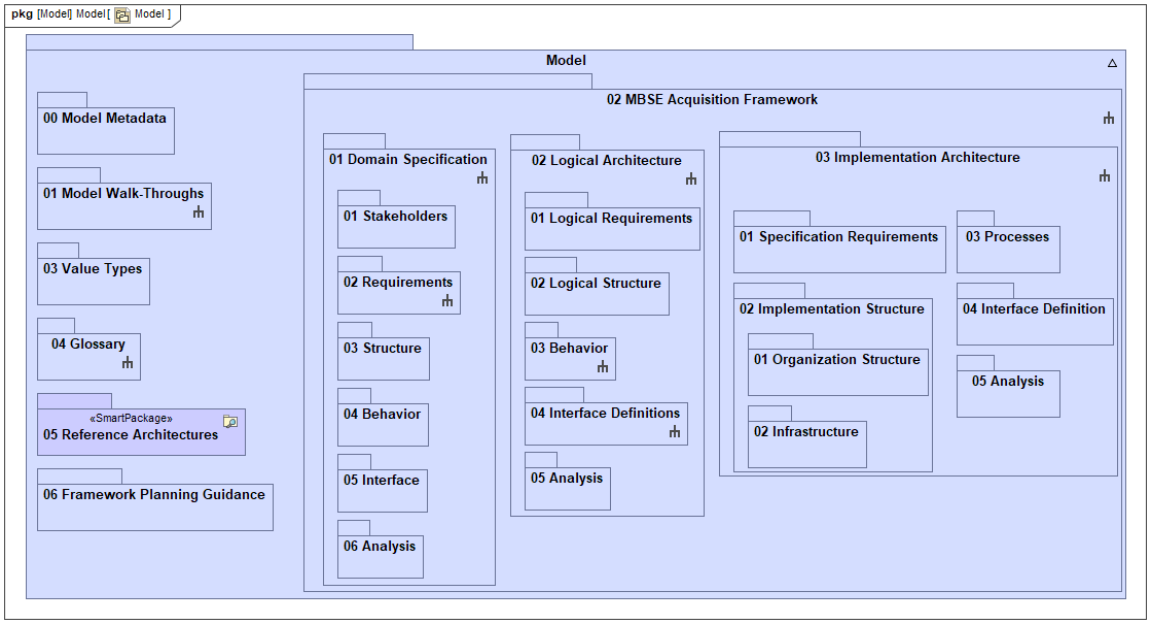

Figure 25. Missile Development Systems Engineering Life Cycle Model Package Structure.

B. MBSE FRAMEWORK STAKEHOLDER NEEDS AND PROCESS REQUIREMENTS

Once planning is complete the next step is to capture stakeholder needs for acquisition framework in which the MBSE framework will be a key portion. This was captured using the process defined in Section III.B. The process created several modeling views and relationships that help to validate the framework meets the needs of the stakeholders.

The team began by identifying the stakeholders who would be involved with the program and identifying both their interest in and their power to influence the MBSE Framework. Stakeholders with both a high interest and a high power to influence were given more attention when creating stakeholder needs. Figure 26 is a depiction of the stakeholders and their power and influence over the framework. The location of the stakeholders on the diagram is not information captured with the SysML standard, so property descriptions were created to document the general location of the stakeholders on the graph. This method is shown with the “MBSE Framework Stakeholder” block from which each block is a specialization. The “Program Office” block is showing the values

for interest and power, but these are hidden on the other blocks for stakeholders for readability. From a modeling methodology, a generalization was used to identify the generic SysML blocks as stakeholders and to inherit the stakeholder interest and stakeholder power property value types. The stakeholders can now be traced to aspects of the model that concern them. For example, the Test Range can be linked to activities and products that concern flight tests.

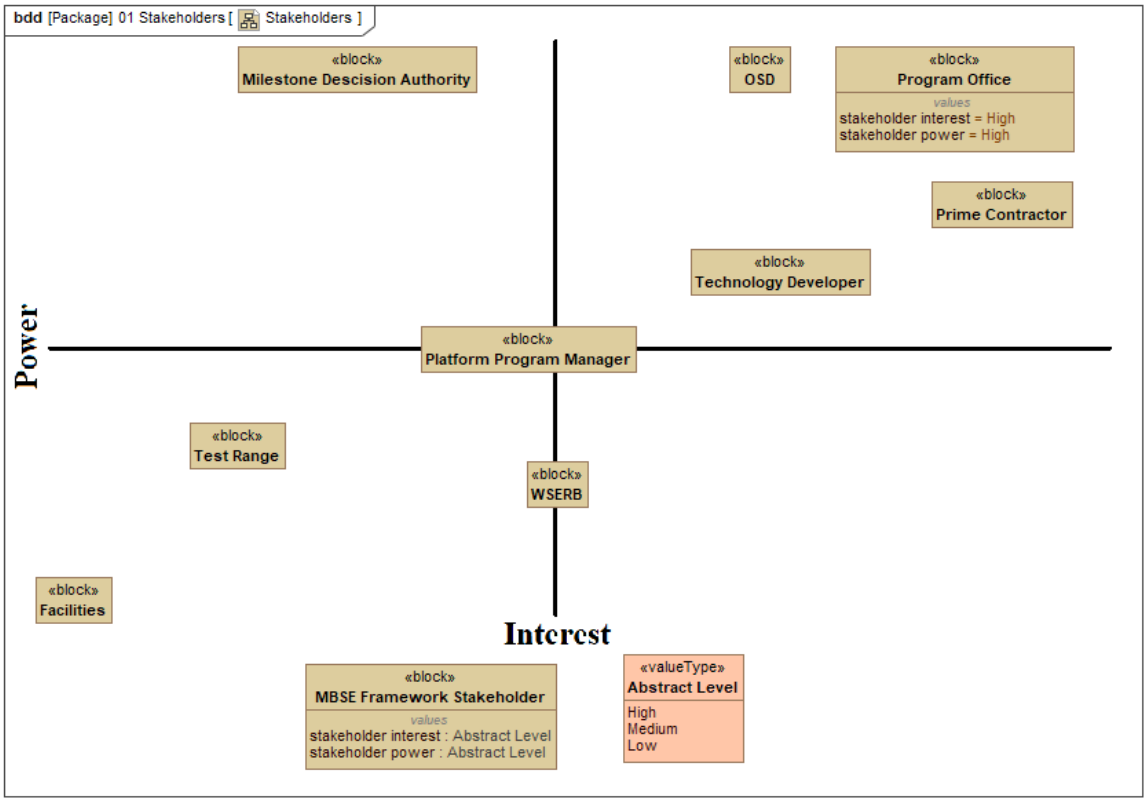

Figure 26. MBSE Framework Stakeholders

The next step was to define a set of stakeholder needs for the programmatic and systems engineering processes. Table 4 identifies a subset of these stakeholder needs. These stakeholder needs are not for the actual system but are instead for the development life cycle of the system. These needs will drive the process towards contract, infrastructure, resource, and process requirements for which the MBSE Framework must meet. The team

kept the list of stakeholder needs simple and expanded upon them in the next step for setting requirements for the MBSE Framework.

Table 4. MBSE Framework Stakeholder Needs

Need Title	Stakeholder Need	Measure of Effectiveness
476 Shorter Time to Deployment	The acquisition life cycle shall provide new missile strike capability in less than 6 years in the fleet.	Successfully fielded prototype in 7 years.
478 Better Government Design Understanding	The government shall understand the contractor design well enough to be able to recompete the build of the system.	Government could recompete contract to a new contractor in 7 years to build with government oversite of production.
479 Higher Design Confidence	The development processes shall have increased confidence in the design at earlier stages of the life cycle.	90% first pass yield on destructive tests.
480 Design Flexibility	The design products shall be flexible to edit for future iterations of the system.	Deployment of second system version with upgrades within 3.5 years of new design capability

The acquisition domain was then captured as seen in Figure 27. The MBSE Framework is the portion of the acquisition development framework that the Program Manager is directly responsible for. The other elements belonging to the acquisition framework in Figure 27 are those that the program must interact with for successful system development. The MBSE Framework is decomposed logically in Section D and with an implementation in Section E. The acquisition domain captures important external actors the development team will interact with in the development of the system that is outside the control of the program office. The model uses the logical item flows of information and resources in Figure 27 to develop additional stakeholder needs and refine already identified stakeholder needs.

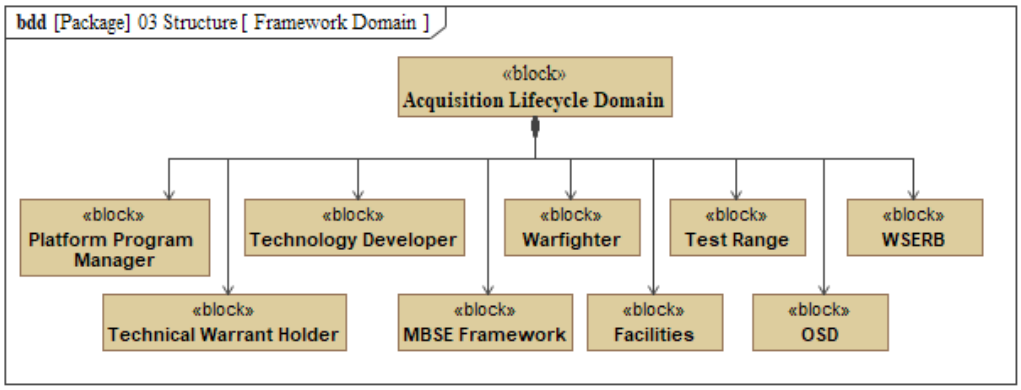

Figure 27. Acquisition Domain

The project then identified the top uses cases and the capabilities that would enable those use cases. Figure 28 shows the top use cases for the MBSE Framework and the relationship of the use cases to the relevant stakeholders. The use cases were defined to be the top-level functions that would enable the stakeholder needs to be met and have a SysML satisfy relationship to the stakeholder needs. The team brainstormed more potential use cases, but use cases can distract from the overall intent of defining a process framework. The project used these use cases as functions of the system with many categorized below those identified in Figure 28 as functions (or actions in SysML).

Figure 29 is an example for defined capabilities that would aid a use case. Some capabilities were a dependency to enabling multiple use cases. For example, real-time access to national team digital products is a dependency for use cases Manage Authoritative Source of Truth and Encourage Artifact Reuse. With a SysML relationship between stakeholder needs and use cases in addition to a modeled relationship between use cases and capabilities, capabilities have a trace to stakeholder needs for analysis.

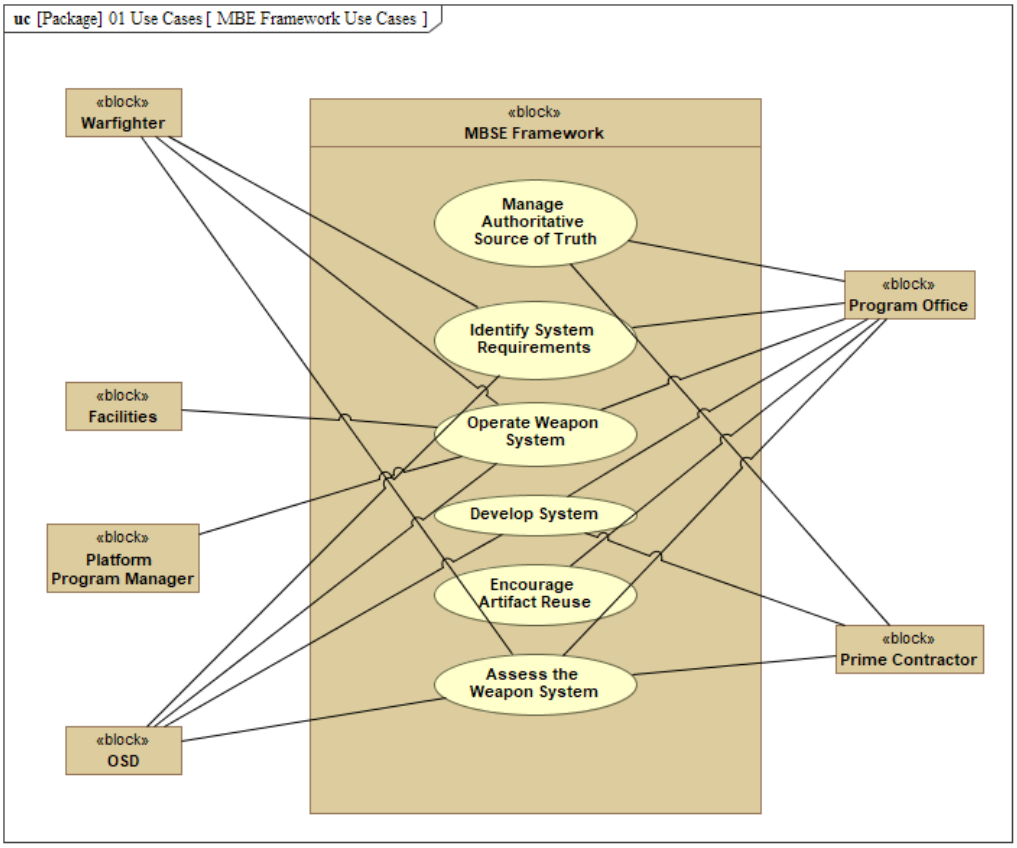

Figure 28. MBSE Framework Use Cases

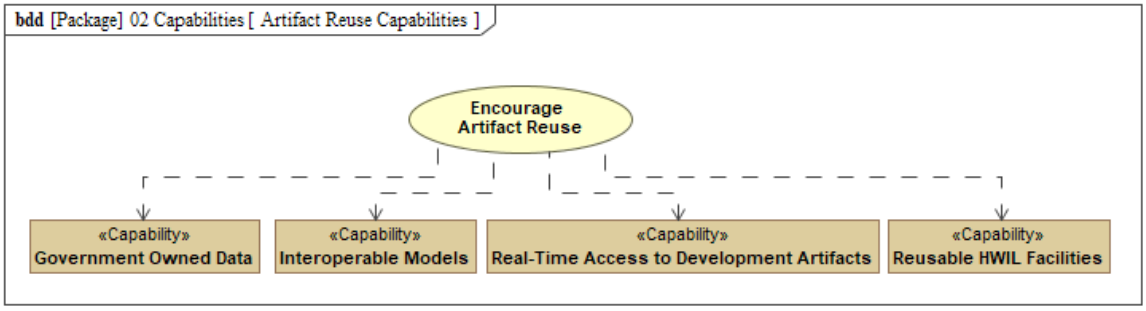

Figure 29. Capabilities to Encourage Artifact Reuse

The last step to defining stakeholder needs was to define Measures of Effectiveness (MoEs) to allow the team to assess the success of the process in the future. These MoEs are verifiable pass/fail criteria once the project is complete. Table 4 captures MoEs for a subset of the stakeholder needs.

C. MBSE FRAMEWORK REQUIREMENT CAPTURE

Requirements were set for the MBSE Framework to set the needs of the framework and to define the interfaces that the MBSE Framework has with external actors. The process from Section III.C was used to define the requirements. The requirements defined in this section are few. The model assumes that external entities would desire traditional looking information and reports on the system development. The majority of the impacts therefore derived from the system will be within the MBSE framework itself in the following sections. As the Navy and The Office of the Secretary of Defense (OSD) adopt MBSE practices, requirements for formats and capabilities of MBSE artifacts will likely grow. However, this section still covers how the interactions to external organizations can be captured, and how a team can translate these interactions into programmatic requirements.

The project first defined scenarios by using SysML sequence diagrams that defined the interactions between the MBSE Framework and the external organizations to the program office. Figure 30 shows the scenario for interactions between the program office inside of the MBSE Framework and OSD. The scenario identifies the information and resources that need to flow between OSD and the MBSE Framework for the program to be successful. The model does not capture off-nominal scenarios, such as a failed flight experiment or a scope change from OSD for the program, for this project. However, any off nominal scenarios should be identified separately from nominal cases.

One item of note from the sequence diagram is the difference with requirements. A traditional program would begin with OSD providing an Initial Capability Document (ICD) to the Program Office, and the Program Office would refine to a Capability Development Document (CDD) (DoD 2018a). Because this program is an MTA, it is not required to go through the traditional Joint Capabilities Integration and Development System (JCIDS) process that would result in a CDD (DoD 2019). This program would use a Top-Level Requirements document in place of a CDD that is more flexible with fewer requirements to allow additional flexibility.

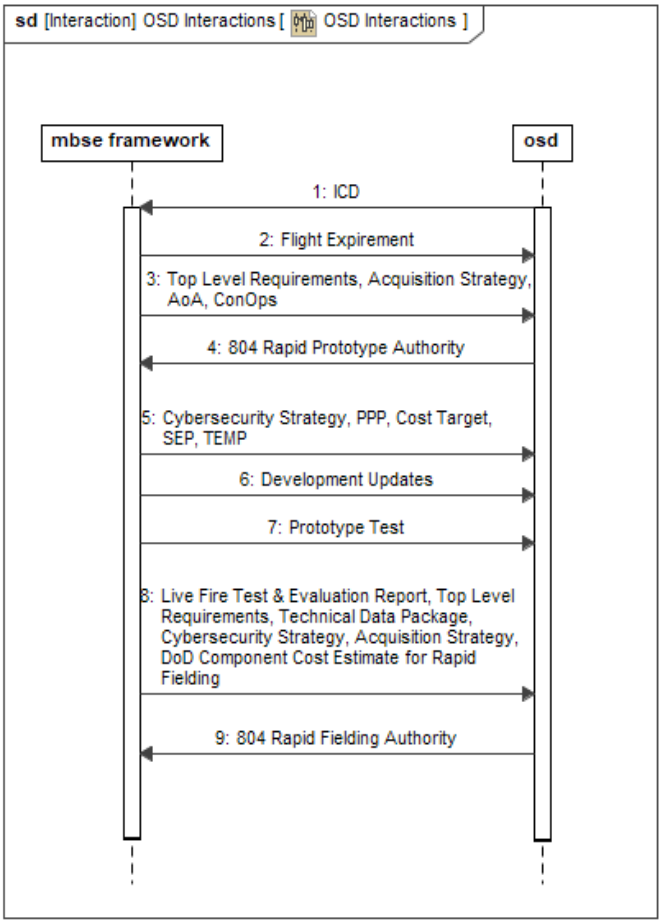

Figure 30. MBSE Framework Scenario for Interactions between OSD and MBSE Framework

Figure 31 shows the system context diagram for the Acquisition Life cycle Framework as identified from the information captured in the sequence scenarios. The system context diagram is a SysML internal block diagram (ibd) that focuses on the information and resource flows between the MBSE Framework and those that will be interacting with the MBSE Framework. The project translates each flow into an interface requirement for the framework to either provide or adhere. The project using an MTA program resulted in fewer interfaces as compared to a traditional MCA program. The project refined the traditional exchanges to only those identified as critical or greatly beneficial.

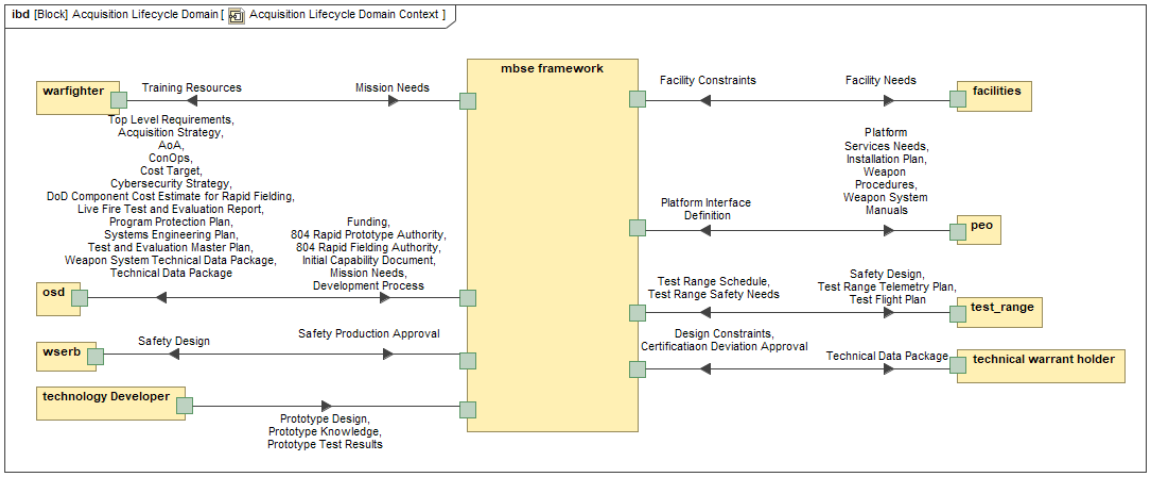

Figure 31. MBSE Framework Context Diagram

The project next identified constraints for the MBSE Framework and Critical Constraints. Because the project used an MTA program, the framework has fewer constraints than would be placed upon an MCA program. However, the MBSE Framework still needs to adhere to acquisition, cybersecurity (for both enterprise and system), safety, test, and other constraints. These constraints still exist without the traditional DoD instructions for MCA being levied on the program. Critical constraints for the MBSE Framework were difficult to identify. Critical constraints should be property based on the system and measurable (Friedenthal 2014), but one is that neither the rapid prototyping nor rapid fielding phases should last more than five years (DoD 2019). Another critical constraint would be the program budget.

The model captures states for the development of the system under the MBSE Framework as Figure 32 illustrates. The states adhere to those defined for an MTA, and the transitions identify why the program would transition to a traditional MCA. The simple states allow flexibility for elements of the system will be at different levels of development. The system may not yet have a complete functional baseline under government control, but elements of the system may already have a government-controlled product baseline. In the case of a missile system, this can be true with the missile having a government controlled product baseline for low rate production for flight tests with experimental hardware while the platform weapon control system does not yet have a government

controlled functional baseline. Further refinement of the development phases can be done each major element of the system through additional states for each element, similar to how subsystems of a system define their own states that adhere to the system's states.

The states also include transitions for how program will transition between states. After planning and successful experimental prototypes, the program would be granted rapid prototyping authority as an MTA program. If the program has a successful flight a fieldable prototype, the program can be granted rapid fielding authority. Once the prototype is fielded, the program would transition to a traditional MCA to further refine the design to extend capabilities and operational life of the system. The program could be moved from a rapid prototyping to a MCA, however, if there is a major setback in the program such as a failed flight experiment, budget issues, schedule delays in which OSD would desire to have the additional oversite that an MCA provides.

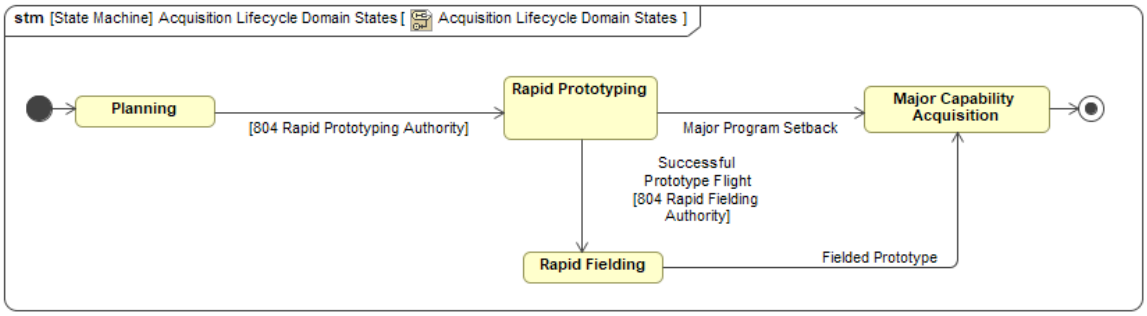

Figure 32. MTA Development States

The last step in this phase for the team was to define the requirements for the MBSE Framework as a black box. The previous steps in identifying interfaces to the external organizations, the constraints that would be placed upon the program, and the critical constraints for performance of the MBSE Framework were used to create the requirements. The project set requirement attributes such as the type of requirement and traceability to other model elements was created. Table 5 provides a subset of the defined requirements. The attributes in the table include the requirement's type and the architecture element(s) that satisfy the requirement.

Table 5. MBSE Framework Example Requirements

Name	Text	Satisfied By	Requirement Type
481 Development Updates	The program shall provide OSD with development updates that include the acquisition strategy updates, system development progress, and analysis from flight test events.	Item Flow:flow for Acquisition Strategy [MBSE Framework -> OSD] Item Flow:flow for Live Fire Test and Evaluation Report [MBSE Framework -> OSD] Item Flow:flow for Systems Engineering Plan [MBSE Framework -> OSD]	interfaceRequirement [Class]
489 Top Level Requirements for Rapid Fielding	The program shall provide OSD with an update of the Top Level Requirements prior to exiting Rapid Prototyping and entering Rapid Fielding.	Item Flow:flow for Top Level Requirements [MBSE Framework -> OSD]	interfaceRequirement [Class]
491 Top Level Requirements	The program shall provide OSD with the Top Level Requirements for submission for 804 Rapid Prototyping Authority.	Item Flow:flow for Top Level Requirements [MBSE Framework -> OSD]	interfaceRequirement [Class]
492 Acquisition Strategy	The program shall provide OSD with the acquisition strategy for the program for submission for 804 Rapid Prototyping Authority.	Item Flow:flow for Acquisition Strategy [MBSE Framework -> OSD]	interfaceRequirement [Class]
490 DISA	The program shall adhere to DISA requirements for enterprise integrated technology infrastructure.	mbse framework : MBSE Framework	designConstraint [Class]

D. MBSE FRAMEWORK LOGICAL ARCHITECTURE

This section examines how the project modeled the logical architecture for the MBSE Framework. Due to the lack of stakeholder requirements for organizations external to the MBSE Framework to receive native MBSE products for this project, the MBSE aspects for developing a system become more relevant in this section. Aspects of the model that deviate from a traditional systems engineering process will be identified.

1. MBSE Framework Logical Structure

The project first identified a structure for the logical MBSE Framework. A WBS is well suited for this structural decomposition of a logical acquisition structure. This project

used MIL-STD-881 *Work Breakdown Structures for Defense Materiel Items* (DoD 2018b) as a starting point for creating the WBS for the MBSE Framework from the example provided for Missile/Ordnance Systems. This project expands upon this example to include MBSE Framework related structure. Figure 33 shows the MBSE Framework WBS with both the added and original structure items that are of interest to an MBSE methodology. The WBS can and should be refined to further detail as a program begins to refine the deliverables that will be expected from both the program office and the contractor(s).

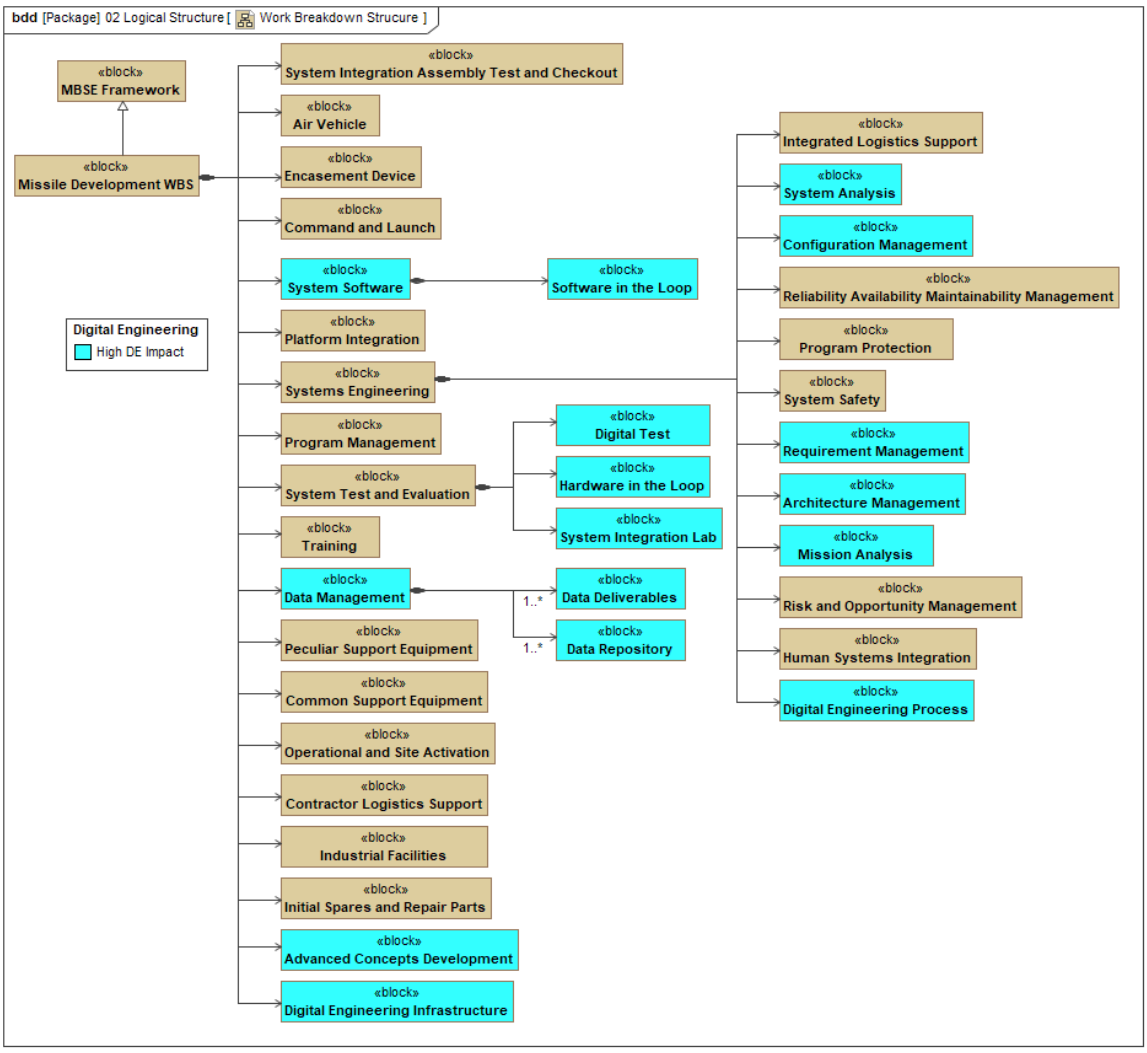

Figure 33. MBSE Framework Work Breakdown Structure

The next step in the project was to create functions, or activities and actions in SysML, for the MBSE Framework. The project initially created functions against the traditional backdrop of the acquisition phases as according to the DAG (DoD 2018a) for an MCA by creating allocations to states that matched the phases from the DAG. Coming from what is known is useful to then modify. The functions were also allocated to the SE processes that were aligned to the *INCOSE Systems Engineering Handbook* (INCOSE 2015) and the DAG (DoD 2018a). The project used the DAG (DoD 2018a) and "Integrating Model Based Engineering, and Trade Space Exploration into Naval Acquisitions" (Stepanchick 2016) as the basis for creating the functions. However, the large number of functions distributed across the allocations make it difficult to read. This project abandoned this method to capture the functionality of the MBSE Framework for an approach that allowed for less complexity in a singular view.

2. MBSE Framework Logical Behavior

The functions of the MBSE Framework are modeled using activity diagrams and the actions are allocated to the WBS structure shown in the previous section. Originally, the functions were created with a more waterfall approach having the majority of the actions on a single diagram showing the interactions between the actions. This method was similar with a traditional project schedule, but it became difficult to manage. The detail was too great for the intent of this project. Instead, many activities were created and allocated to the MBSE Framework block. This simplified the modeling, but still allowed for the model to explore the decomposition of the MBSE Framework process and the allocation to the WBS. This section includes some example activities that are most relevant to MBSE from that work and how the actions were allocated to the WBS.

Figure 34 provides an example activity diagram for the development of an architecture for a system with allocations to the WBS elements that will be the lead in its creation. Allocations are to the lead because many subprocesses will have other portions of the WBS involved. For instance, the system integration WBS element is the lead for "design system architecture," but the test, which is not structurally under system integration, will be involved with the architecture when developing test cases for the

system. The actions are based on the first Vee from “Integrating Model Based Engineering and Trade Space Exploration into Naval Acquisitions,” except the below includes requirements within the architecture model (Stepanchick 2016). The verification step is added to verify the architecture model can be read correctly for validation, trade study, and integration into SoS models actions.

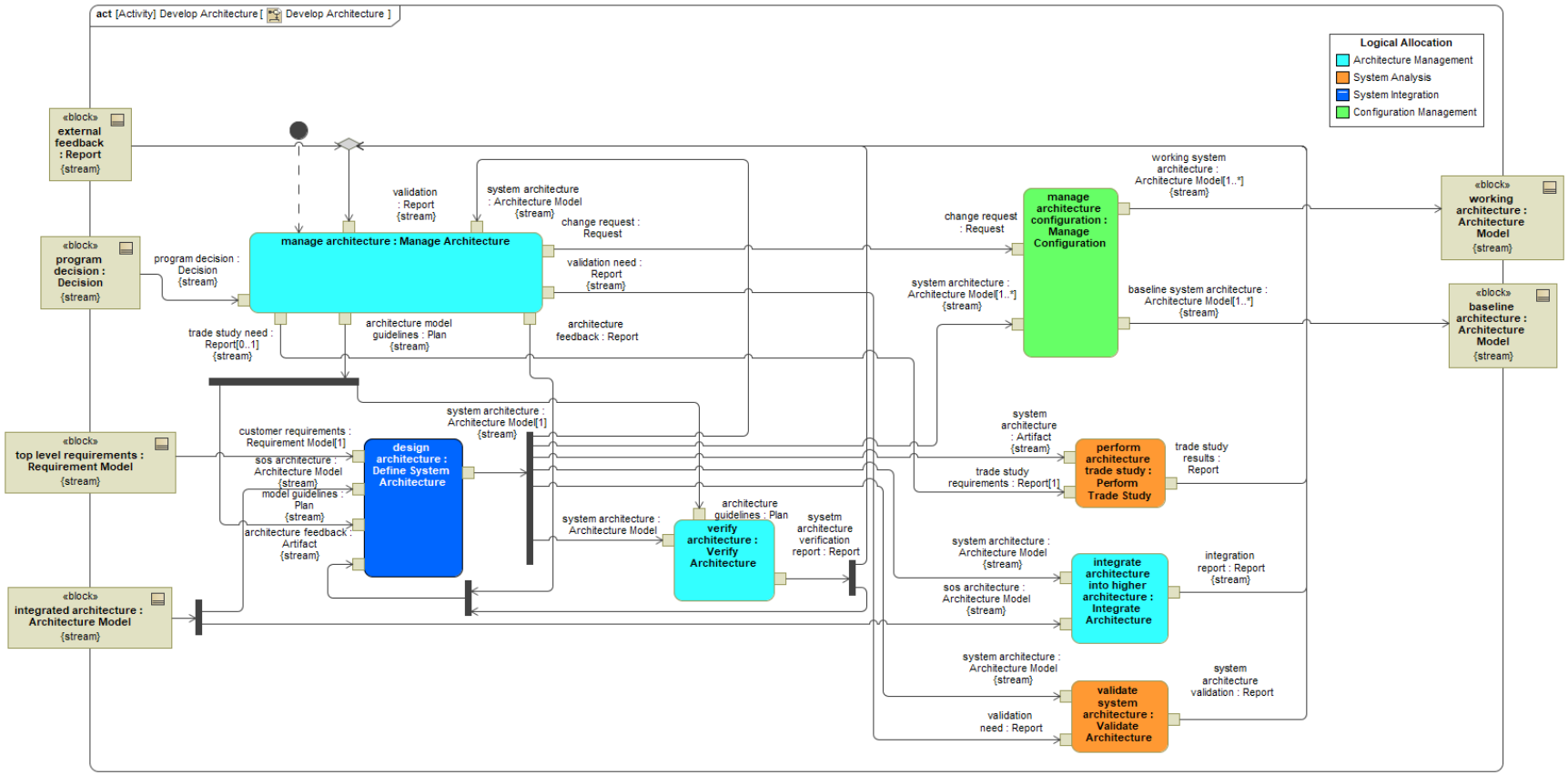

Figure 34. Activity Design Architecture with Logical Allocations

Some key SysML aspects are used in building the activities that the MBSE Framework performs. One key is that many ports on actions have the SysML standard <<stream>> stereotype applied. A streaming parameter allows the action to continue to receive and output additional information as the action is active, as opposed to an action that would start running with current information and only output information upon completion (Friedenthal, Moore, and Steiner 2014, 210). This is important for a program that wants to include agile development with continuous updates to the current execution plan. Actions with all stream inputs and outputs such as “verify architecture” are expected to be continuous functions. Actions that do not have streaming inputs such as “validate system architecture” are expected to be functions that do not occur continuously but are kicked off as needed. Verification of the system architecture in a MBSE Framework is expected to be an automated similar to software testing, but validation of the system architecture is expected to require manual review and specialized analysis as required by the analysis team.

Another SysML method used is to reuse activities to classify actions. An example is the “Manage Configuration” activity reused as several actions. The activity has parameter nodes with type “Artifact”, but the actions can be redefined with a more specific element input as long as the element is a specialization of “Artifact”. A taxonomy of development artifacts with more specific elements created with the SysML generalization relationship as seen in Figure 36 allows reuse of the “Manage Configuration” activity throughout the model without recreating additional activities. Figure 36 demonstrates this capability with the “Manage Configuration” activity being used to classify three different actions with different types of inputs and outputs. This speeds development of the model and helps manage changes to the model if configuration management processes change.

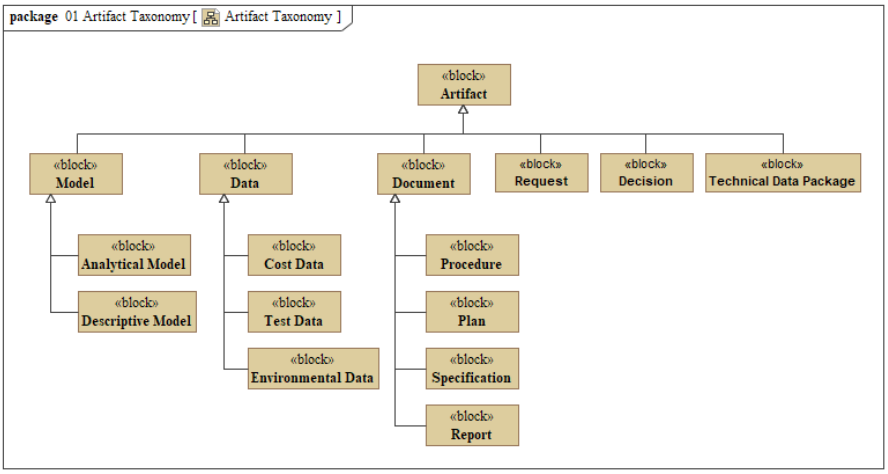

Figure 35. Artifact Taxonomy for Interfaces

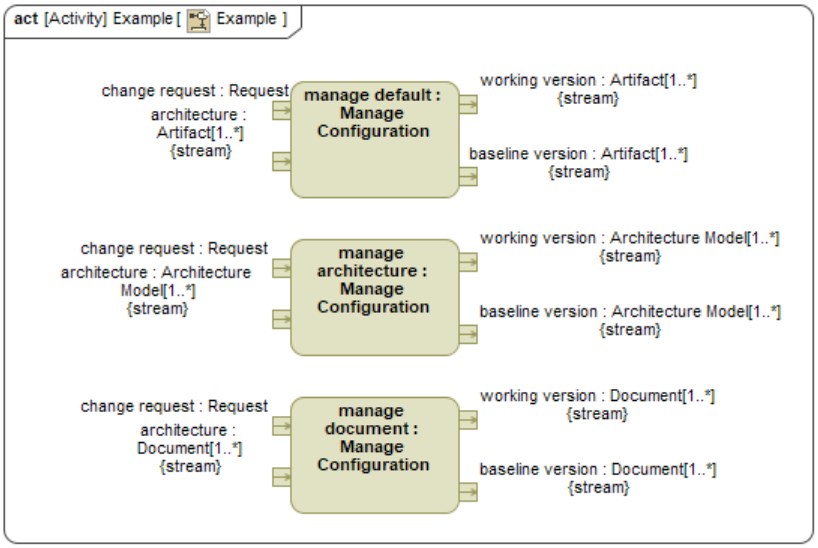

Figure 36. SysML Action Polymorphism with Manage Configuration Example

Figure 37 provides the continuation of development of the system with developing the digital design of the system. This activity would include designing the functional implementation represented by functional simulations that developers will implement with hardware or software elements. The functional simulations represent the performance of both hardware and software elements (Stepanchick 2016). The activities completed within the actions align with the activities in Figure 10 from Stepanchick (2016) with his "Vee"

iteration 3 within the Digital Domain, except System simulation. As with the development of the architecture, the “analyze functional design” action verifies validates both the models themselves and the design captured by the models. Results are fed back to the developers in addition to feedback from external reviews. An example for an external review would be a SETR event or a program protection review of the design that would be occurring in parallel.

Figure 38 depicts functions to verify the system. These actions are align with those in the iteration 3 “Vee” in Figure 10 from Stepanchick (2016) and also includes system simulation. “verify digital design” would be verifying the purely digital models created for the system. “verify design” would use HWIL, SWIL, physical mockups, virtual reality, analytical analysis, and other techniques that can be used to build confidence in the system using a mixture of manufactured components and models to increase confidence the design will meet requirements prior to full integration. “verify system” would take the results from the design verification and assembly verification and use them in conjunction with weapon system tests at test ranges and other test facilities to verify the system. The results of the system verification tests are then used to verify the design models to ensure they are accurate. This step is important because results from the design verification are used to run additional test cases that are not affordable through test flights. The models will almost certainly need to be update with data from tests of the real system to ensure they are accurate. The last function to mention is “accredit model” to formally allow the models to be used to verify requirements and scope the requirements the models can verify.

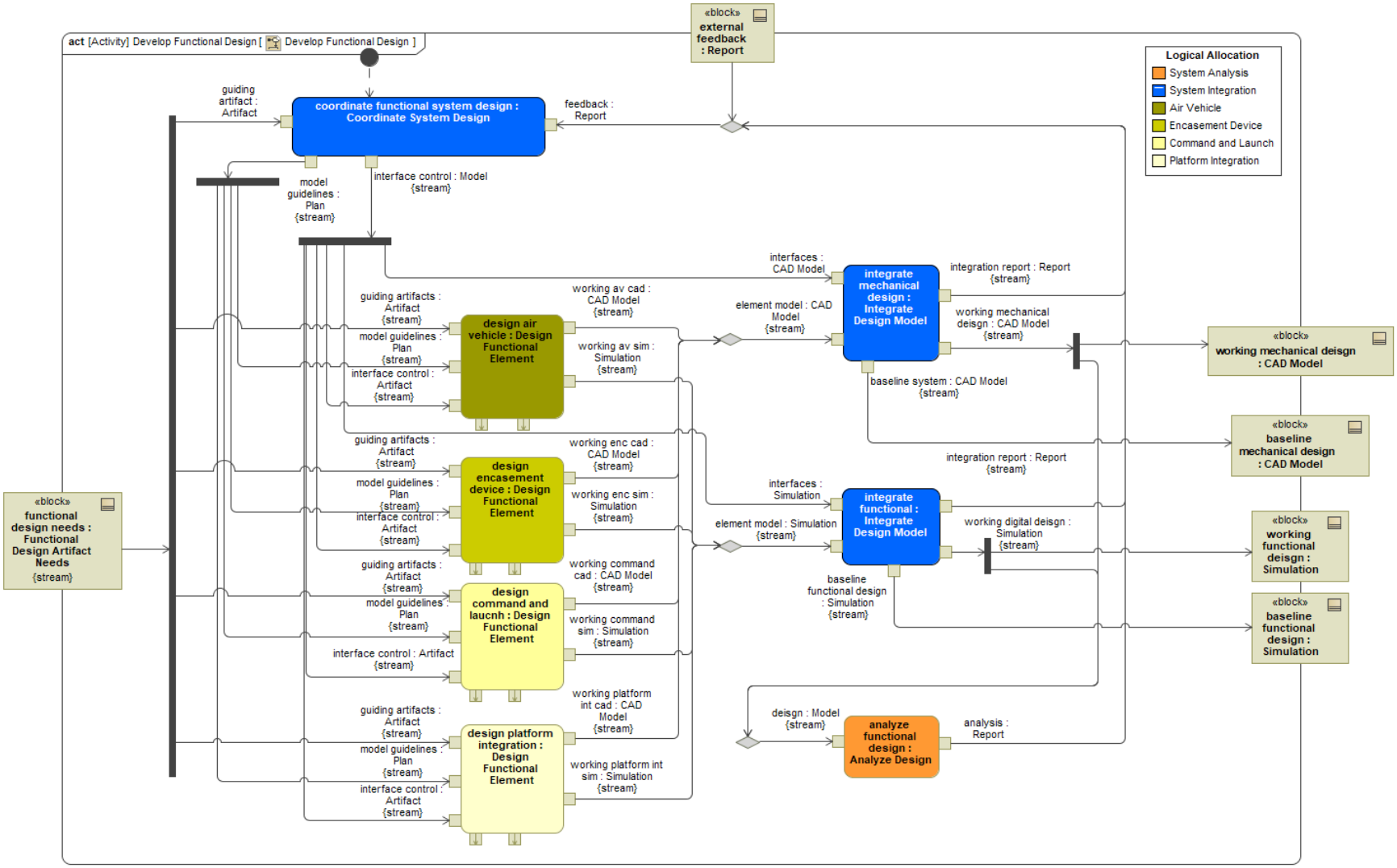

Figure 37. Activity Design Digital System with Logical Allocations

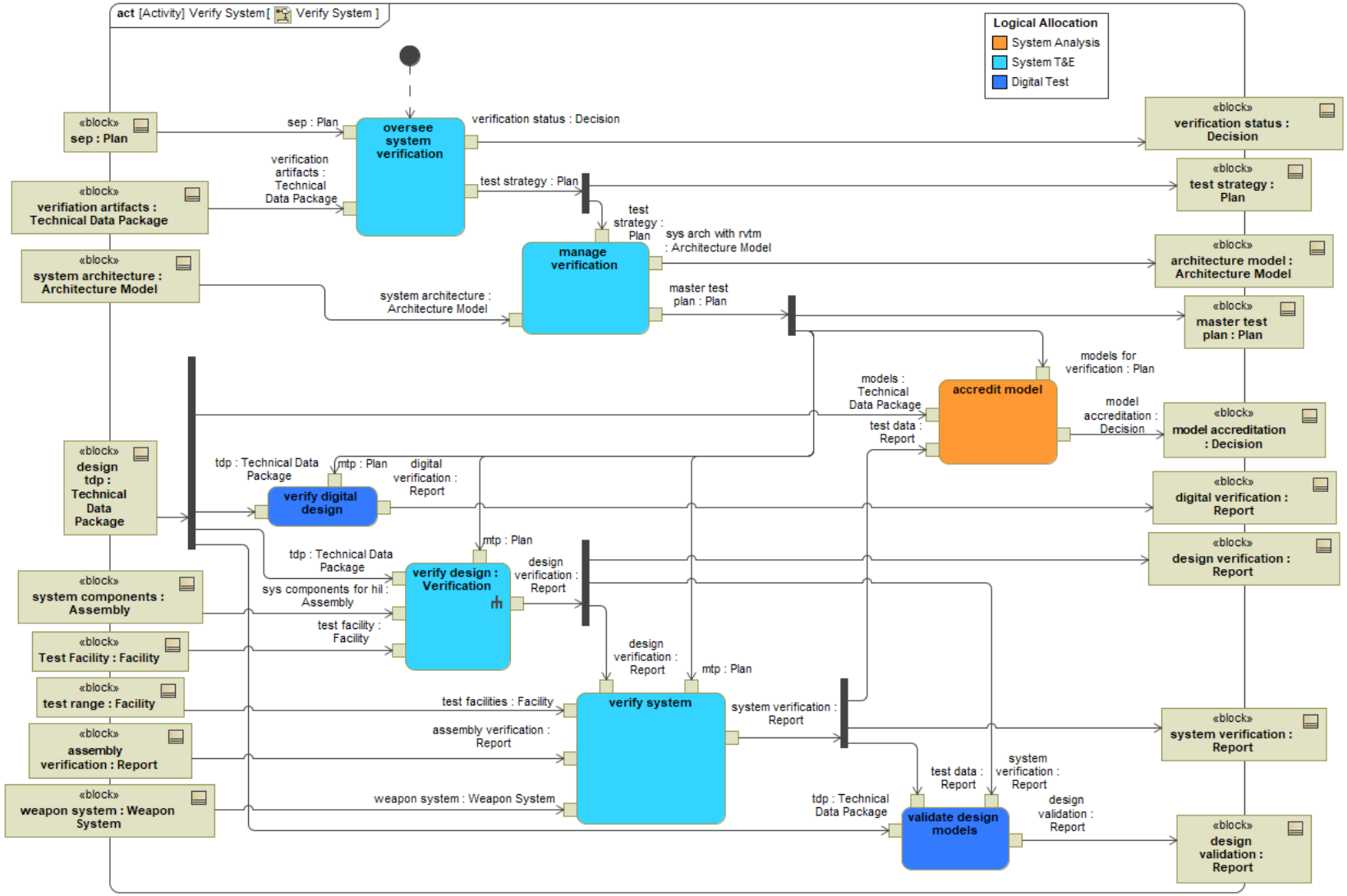

Figure 38. Activity Verify System with Logical Allocations

Figure 37 uses an input of type “Functional Design Artifact Needs” to represent the documents, models, and other artifacts that would be used to design the functional digital system. This is used to simplify the diagram from the requirement to show each artifact separately. Figure 39 shows the decomposition of the “Functional Design Artifact Needs,” and Table 6 shows a simpler view of the same information in table format from data pulled from the model with an addition of showing the description of the items. The table will automatically update if the model elements are changed and is a better view for model readers.

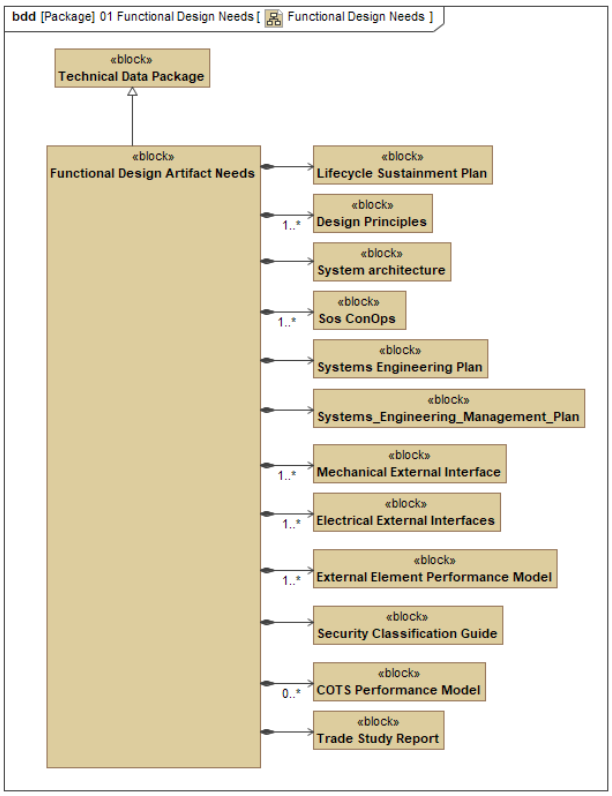

Figure 39. Technical Data Package Elements for Input to Develop Functional Design

Table 6. Technical Data Package Elements for Input to Develop Functional Design

#	Type	Multiplicity	Documentation
1	Internal Element Performance Model	0..*	Reusable performance model for an element of the system that was developed outside the context of the current development.
2	Design Principles	1..*	Guiding documents such as mil standards, company documents, or program specific plans that guide the development of the system.
3	Electrical External Interfaces	1..*	Electrical, including wireless communication, interfaces that the system will integrate with on a platform or other external systems (e.g. GPS).
4	External Element Performance Model	1..*	Simulations or other analytical models for systems external to the system that impact the performance of the weapon system of interest under development
5	Lifecycle Sustainment Plan	(Unspecified)	Plan for the logistic activities to be conducted throughout the life of the system
6	Mechanical External Interface	(Unspecified)	Mechanical interfaces that the system will integrate with on a platform or other external systems (e.g. common support equipment).
7	Security Classification Guide	(Unspecified)	Guidance for classification level of program information.
8	Systems Engineering Management Plan	(Unspecified)	Plan created by the prime contractor for how they will develop the system in accordance with the Systems Engineering
9	Systems Engineering Plan	(Unspecified)	Plan for how the program develop the weapon system.
10	Sos ConOps	(Unspecified)	Platform, mission, logistics or other models that the weapon system will integrate with.
11	System Architecture	(Unspecified)	Architecture model for the system of interest.
12	Trade Study Report	(Unspecified)	Trade studies conducted that can influence the design of the system.

3. MBSE Framework Logical Behavior Mapping to Current Process Behavior

Part of the process of defining activities is to link them to current processes. The example provided here is a trace to the *INCOSE Systems Engineering Handbook* (INCOSE 2015) as a current process was not modeled. This trace was also done for the processes from the DAG (DoD 2017a). Figure 40 shows a matrix with the actions from "Develop Architecture" traced to *INCOSE Systems Engineering Handbook* (INCOSE 2015) functions using a SysML Abstraction relationship. Tables can then be generated for

reference to pull information that is beneficial to those that will need to perform the function or for the purposes of decomposing the function. The information could be traces to activity diagrams from a previously modeled process or as Table 7 does by pulling in the documentation for the function. The documentation in Table 7 is from the *INCOSE Systems Engineering Handbook* (INCOSE 2015).

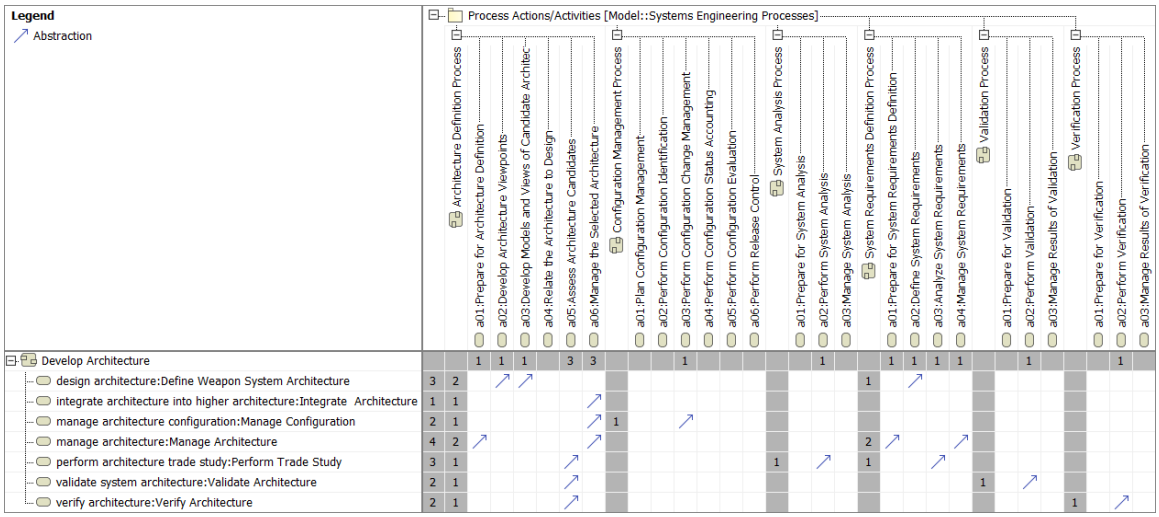

Figure 40. Mapping of Logical Architecture Process to INCOSE SE Processes. Adapted from INCOSE (2015).

Table 7. INCOSE Documentation for Design Architecture Action. Adapted from INCOSE (2015, 59, 65–67).

#	Name	INCOSE function description
1	design architecture	***4.4.1.4 Process Activities*** *Develop architecture viewpoints.* - Based on the identified stakeholder concerns, establish or identify the associated architecture viewpoints, the supporting kinds of models that facilitate the analysis and understanding of the viewpoint, and relevant architecture frameworks to support the development of the models and views.
		4.4.1.4 Process Activities *Develop models and views of candidate architectures.* - Select or develop supporting modeling techniques and tools. - In conjunction with the system requirements definition process, determine the system context (i.e., how the SOI fits into the external environment) and boundary, including the interfaces, that reflect the operational scenarios and expected system behaviors. This task includes identification of expected interactions of the system with systems or other entities external to the system (control) boundary as defined in negotiated ICDs. - Determine which architectural entities (e.g., functions, input/output flows, system elements, physical interfaces, architectural characteristics, information/data elements, containers, nodes, links, communication resources, etc.) address the highest priority requirements (i.e., most important stakeholder concerns, critical quality characteristics, and other critical needs). - Allocate concepts, properties, characteristics, behaviors, functions, and/or constraints that are significant to architecture decisions of the system to architectural entities. - Select, adapt, or develop models of the candidate architectures of the system, such as logical and physical models. It is sometimes neither necessary nor sufficient to use logical and physical models. The models to be used are those that best address key stakeholder concerns. Logical models may include functional, behavioral, or temporal models; physical models may include structural blocks, mass, layout, and other physical models (see Section 9.1 for more information about models). - Determine need for derived system requirements induced by necessary added architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions). Use the system requirements definition process to define and formalize them. - Compose views from the models of the candidate architectures. The views are intended to ensure that the stakeholder concerns and critical requirements have been addressed. - For each system element that composes the system, develop requirements corresponding to allocation, alignment, and partitioning of architectural entities and system requirements to system elements. To do this, invoke the stakeholder needs and requirements definition process and the system requirements definition process. - Analyze the architecture models and views for consistency and resolve any issues identified. Correspondence rules from frameworks can be useful in this analysis (ISO/IEC/IEEE 42010, 2011). - Verify and validate the models by execution or simulation, if modeling techniques and tools permit, and with traceability matrix of OpsCon. Where possible, use design tools to check their feasibility and validity. As needed, implement partial mock-ups or prototypes, or use executable architecture prototypes or simulators.
		4.3.1.4 Process Activities *Define system requirements.* - Identify and define the required system functions. These functions should be kept implementation independent, not imposing additional design constraints. Define conditions or design factors that facilitate and foster efficient and cost-effective life cycle functions (e.g., acquisition, deployment, operation, support, and retirement). Also, include the system behavior characteristics. - Identify the stakeholder requirements or organizational limitations that impose unavoidable constraints on the system and capture those constraints. - Identify the critical quality characteristics that are relevant to the system, such as safety, security, reliability, and supportability. - Identify the technical risks that need to be accounted for in the system requirements. - Specify system requirements, consistent with stakeholder requirements, functional boundaries, functions, constraints, critical performance measures, critical quality characteristics, and risks. System requirements may be captured in the SyRS. Additionally, a documentation tree may be developed to define the hierarchy of system definition products being developed. The documentation tree evolves with the interaction of the system requirements definition, architecture definition, and design definition processes. Capture the associated rationale, as the requirements are specified.

4. MBSE Framework Logical Interfaces

The usual method to manage interfaces in SysML is to develop ibds that are used to define proxy ports on parts, connect these ports between parts of the system, and then define interface flows. For this project, an alternative method was used because it met the needs of modeling project to identify the information that flows in and out of each WBS element. A data query of the actions defined in activity diagrams was used to capture the inputs and outputs from each WBS element. Table 8 is an example of these queries scoped to only the actions shown in Figure 34. The query looks for object nodes on actions and ensures input elements are not also an output element. The method to do this type of query

is shown in Section F. As activities or allocations change, the tables are automatically updated. The interfaces between WBS elements can be examined with these tables and the number of elements being exchange summed to evaluate a change in allocation or a change in the process.

Table 8. Logical Architecture Allocation Example for Develop Architecture

#	△ Name	Functions	Inputs	Outputs
1	architecture management	manage architecture:Manage Architecture verify architecture:Verify Architecture integrate architecture into higher architecture:Integrate Architecture	system architecture validation program decision sos architecture	trade study need architecture model guidelines validation need change request architecture feedback sysetm architecture verification re integration report
2	configuration management	manage architecture configuration:Manage Configuration	system architecture architecture change request	working system architecture baseline system architecture
3	system analysis	perform architecture trade study:Perform Trade Study validate system architecture:Validate Architecture	system architecture architecture trade study requ validation need	trade study results system architecture validation
4	system integration	design architecture:Define System Architecture	customer requirements architecture feedback sos architecture model guidelines	system architecture

5. MBSE Framework Logical Digital System Model

Programs use technical data packages (TDP) to receive information from a Prime Contractor to be able to analyze, sustain, and reproduce systems. MIL-STD 31000 (DoD 2018c) defines a TDP as below:

> The authoritative technical description of an item. This technical description supports the acquisition, production, inspection, engineering, and logistics support of the item. The description defines the required design configuration and/or performance requirements, and procedures required to ensure adequacy of item performance. It consists of applicable technical data such as models, engineering design data, associated lists, specifications, standards, performance requirements, quality assurance provisions, software documentation and packaging details. (DoD 2018c)

The DSM is the TDP but with concerns for how the information is defined, what information exists in multiple artifacts, how is it synchronized between artifacts, and which artifact is the source of truth if there is a discrepancy. These concerns can be captured in the model.

For the logical architecture, this begins with mapping what tools will need to handle which types of information elements have been defined in the taxonomy. This was done in

the model using a dependency relationship between the elements identified in Figure 35 and a decomposition of the WBS element for "Digital Engineering Infrastructure". Figure 41 shows a subset of these relationships. The tools along the top of Figure 41 are all logical at this point with no direction as to the commercial or custom tools that will be selected for them. For some of the tools, multiple tools will need to be identified for implementation. An example is the "simulation tool." There is likely not to be a tool that can do all continuous simulation and computational fluid dynamics (CFD) needs, and the program would need to select multiple tools to meet that requirement.

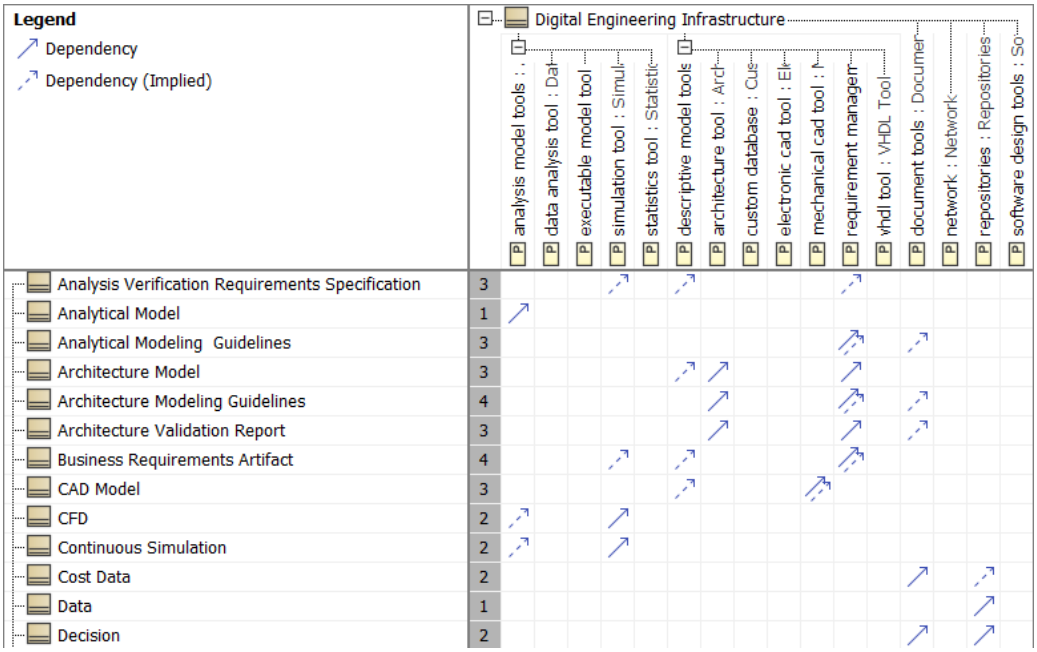

Figure 41. Allocation of Digital Taxonomy

E. MBSE FRAMEWORK IMPLEMENTATION

Once an initial logical architecture is completed, the physical architecture, or the implementation of the MBSE Framework, can be modeled. The implementation consists of the actual organizations that will be supporting the program, the supporting facilities and infrastructure, and the processes that the team will use. This process would be iterative with the logical architecture. Changes to the logical architecture will be identified and should be implemented as the implementation is modeled.

1. MBSE Framework Logical to Implementation Mapping Criteria

The first step to translating the logical structure to the physical structure was for the project to define the criteria used to group the logical structure to the physical structure. This can be a complex set of criteria such as:

- The experience of organizations to complete the functions allocated to the WBS element. Many organizations can leverage experience from previous programs or projects and leverage existing personnel resources.
- The cost for the organization to complete the task. The costs for a government agent to complete will likely be higher than a contractor.
- The ability for the government to exert additional control. The government may want to retain some WBS internally in order to be able to have more direct control of the outcomes and benefit from system knowledge gained within the workforce to continue development of the system in the future.

This project focused less on the former two criteria above as they are more specific to a given project and the organizations vying for the workload. The latter was concerned, however, in assigning tasks between the government and contractors. An example is requirement and architecture management that was desired to be kept inside the government even though WBS elements for integration were still allocated to the prime contractor. The team also used more architecture-specific criteria to reduce the number of interfaces between the organizations. These criteria reduce the complexity of the MBSE Framework and limit unnecessary interactions between potentially geographically disparate organizations.

2. MBSE Framework Implementation Structure

Figure 42 shows the MBSE Framework as the project captured. The decomposition has the different types of organizations that support the program and a fourth component for a digital infrastructure to support the program. The digital infrastructure is separate due to its importance to provide a collaborative environment in which each organization works, set standards for digital products, and to provide facilities that allow for early testing of

prototyped components before a flight test. Figure 43 shows a further decomposition of the program office that demonstrates the object-oriented capability of SysML to only have a singular block represent several roles within the program office. Allocations from the logical architecture is part to part, so each role is given specialized allocations.

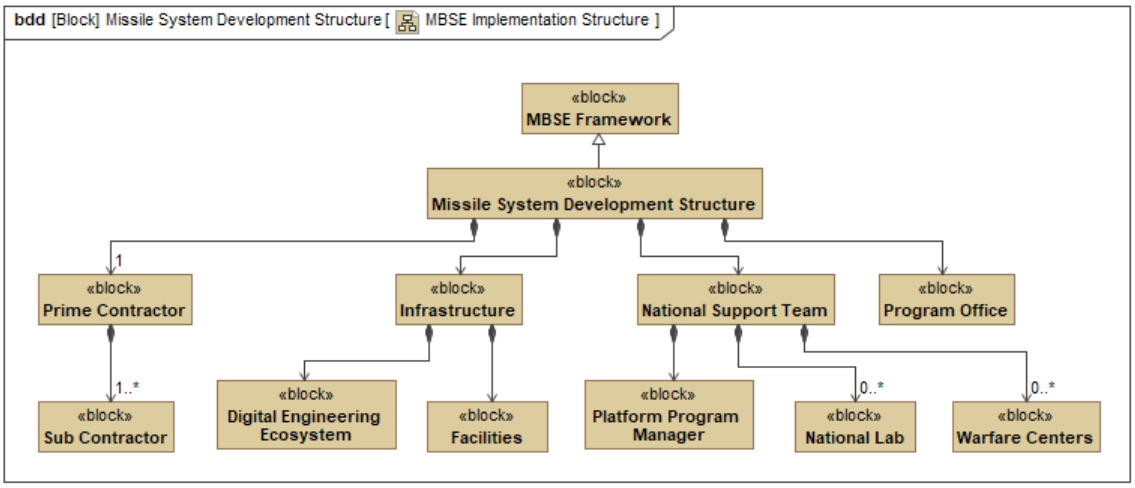

Figure 42. MBSE Framework

3. MBSE Framework Implementation Behavior

New behaviors were not created for the implementation of the MBSE Framework. Instead, the previous behavior was used and updated to with allocations to the implementation structure as opposed to the WBS. Figure 44 displays an example of these new allocations and is the same view as the logical activity shown in Figure 38. This is also where iteration between the logical and implementation definitions is important. The object nodes at this point can also be typed by more specific types. Unfortunately, this will also update the object nodes in the logical views. Figure 38 was created before updating the action object nodes and activity parameter nodes, but Figure 44 has these updates. An example of the change is with the SEP on the parameter node. The type of the parameter node has been updated to the SEP and a number has been given for the name of the parameter that would be the document number for the SEP. The “Systems Engineering Plan” Block is a specialization of a “Plan” Block to allow it to more easily be inserted as a new type and allows for the relationships on the “Plan” Block from the logical model to be

inherited. These more specific types have additional uses that will be explored in a later section.

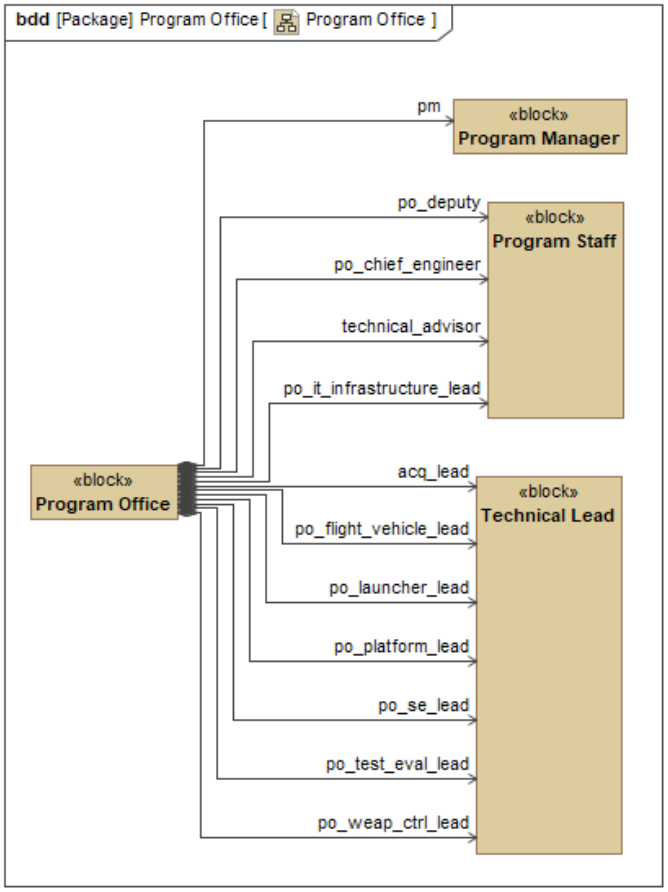

Figure 43. Program Office Decomposition

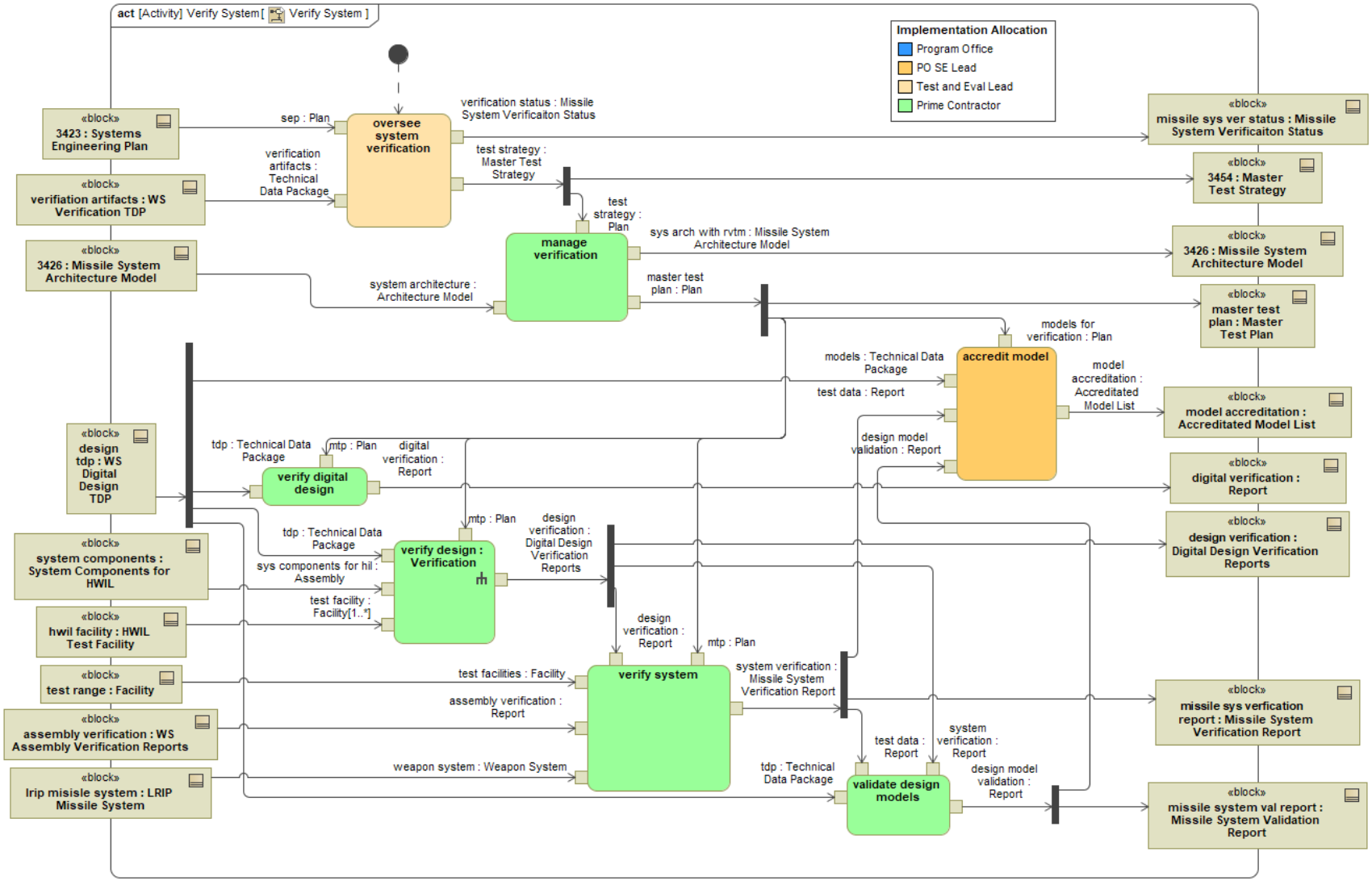

Figure 44. Activity Verify System with Implementation Allocations

4. MBSE Framework Implementation to Logical Mapping

The tracing of the implementation to the logical WBS was done through a data query of the model. In the previous section, the allocation of implementation elements was already discussed. With allocations from the actions to both an implementation structure part and a logical WBS structure part, a query was set up to identify the parts that had the same allocations. The result is seen in Figure 45 as a matrix with direct relationships as solid arrows and relationships to an owned part with a dashed arrow. For each element of the WBS that is traced to more than one implementation element, either the function is being co-chaired with shared responsibilities, or the WBS should be further decomposed. An example of a co-chaired activity might be a joint industry and government working group or a SETR event. As a program matures the activities that need to be accomplished, and splits the duties between the program office, government partners, and the prime contractor, the WBS will need to be further decomposed. This is another example of the iterative nature between defining the logical and implementation of the MBSE Framework.

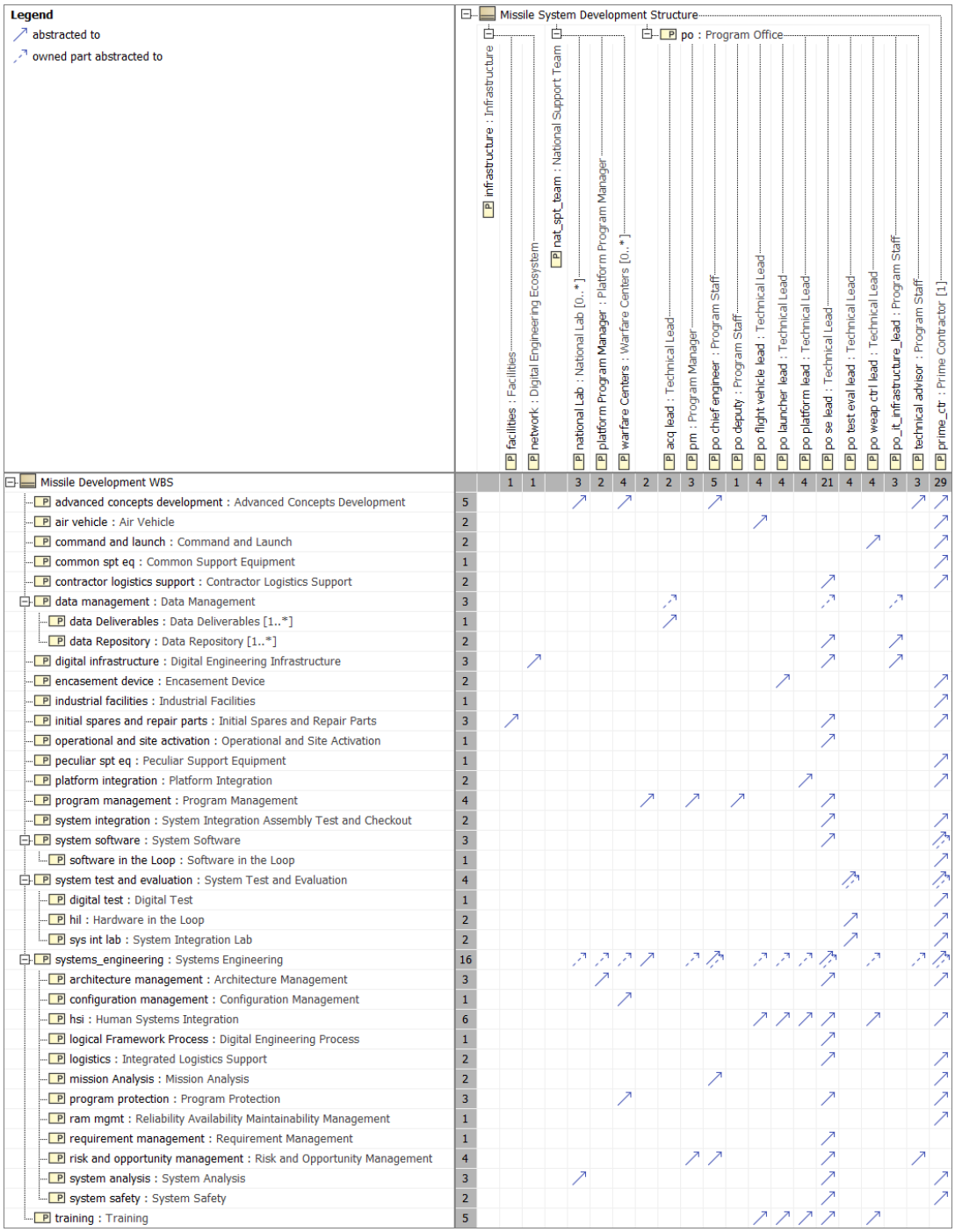

Figure 45. MBSE Framework Implementation to Logical Structure Mapping

5. MBSE Framework Implementation Interfaces

Interfaces can initially be identified with the same methods as used for the logical architecture. Table 9 shows an example of table for the interfaces within the “Develop Architecture” activity.

Table 9. Implementation Architecture Allocation Example for Develop Architecture

#	△ Name	Functions	Inputs	Outputs
1	national Lab	perform architecture trade study:Perform Trade Study	system architecture trade study requirements	trade study results
2	platform Program Manager	integrate architecture into higher architecture:Integrate Architecture	sos architecture system architecture	integration report
3	po_se_lead	manage architecture:Manage Architecture	system architecture validation program decision	trade study need architecture model guidelines validation need change request architecture feedback
4	prime_ctr	perform architecture trade study:Perform Trade Study design architecture:Define System Architecture verify architecture:Verify Architecture	architecture guidelines sos architecture trade study requirements customer requirements architecture feedback model guidelines	trade study results system architecture sysetm architecture verification re
5	warfare Centers	manage architecture configuration:Manage Configuration	system architecture change request	working system architecture baseline system architecture

As the pins of actions are refined for the implementation as was done for the “Verify System” activity, however, the data to be referenced will be the type for the action pins. As this expands over time, the elements that will be coming from the prime contractor can be identified. This can help to formulate the Contract Data Requirements List (CDRL) or set expected entrance criteria at a SETR event for information to be provided to the government. Table 10 is an example of this query for only the actions from Figure 44. The table includes every output from Prime Contractor allocated action, and then also the function from which the element is created or modified.

Table 10. Prime Contractor Developed Artifacts

#	Name	Functions that Output
1	Digital Design Verification Report	verify digital design
2	Missile System Architecture Model	manage verification
3	Missile System Verification Report	verify system
4	Master Test Plan	manage verification
5	Missile System Validation Report	validate design models
6	Design Verification Reports	verify design:Verification

6. Implementation Digital System Model

The previous sections for modeling the implementation focused on the functions and structure of the organizations that execute the functions. The DSM can be greatly expanded upon from the logical implementation to better understand the products the government will manage or what the government wants to request from the Prime Contractor. Figure 46 is an example showing how the ontology for the SEP is modeled. Several extended relationships are shown for the purposes of understanding how the information is gathered. In this example, the SEP is a combination of both a document that is an export of a SysML model and the SysML model itself. An MBSE Framework that is similar to the example modeled in this chapter is the basis of the SEP. The MBSE Framework model can have also be the basis of other elements, so a directed aggregation is used as opposed to a directed composition relationship.

Data queries as those shown in previous sections can allow the data model to be viewed in many different ways. The specializations allow the data to be queried by ensuring common relationships. The values of many elements are inherited through specialization relationships, and some of these inherited values are then redefined for that particular element.

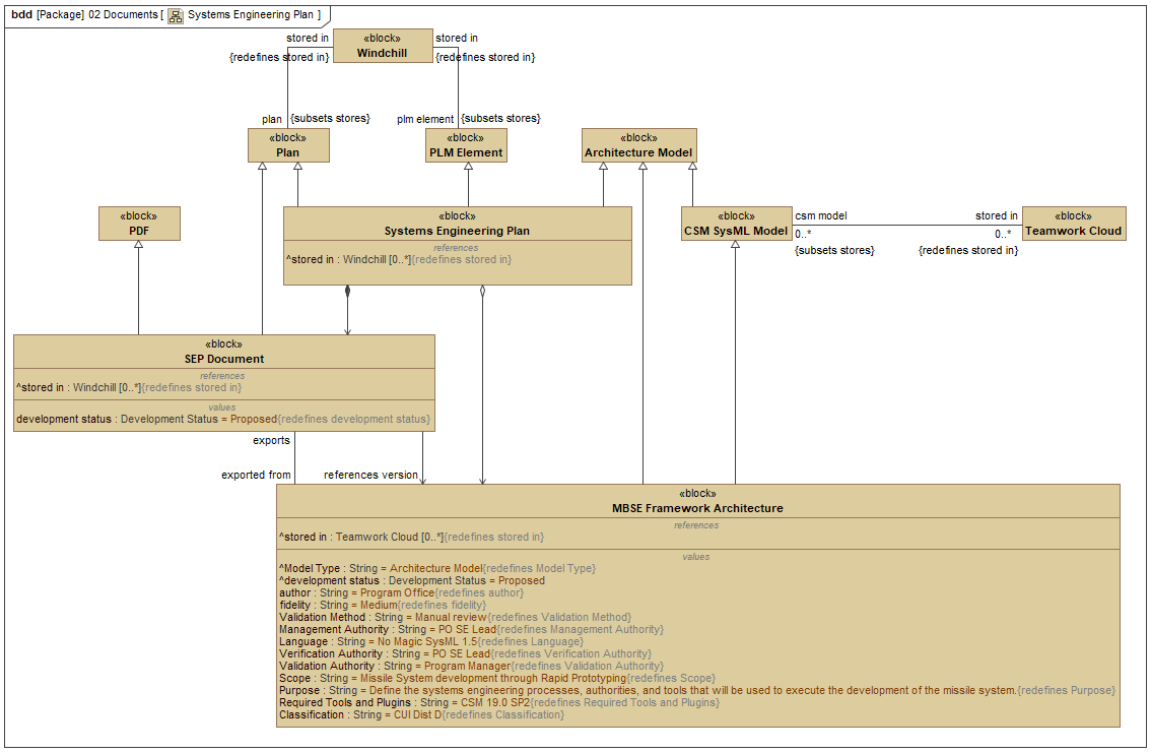

Figure 46. Ontology for Systems Engineering Plan

The artifacts are also traced to the format that they will be in. In this case, the SEP Document is a PDF stored in Windchill and the MBSE Framework Architecture is a SysML model in Cameo Systems Modeler stored in Teamwork Cloud. The SEP itself then is a Product Lifecycle Manager (PLM) element that is stored within Windchill that references both the document and the plan. By capturing this information, the data management team can plan for the infrastructure required to manage the program's data and also get an understanding of the relationships between data sources.

F. EVALUATE AND MAINTAIN MBSE FRAMEWORK MODEL

Partial analysis was conducted on the model to evaluate its completeness through manual review, alternative views of data, and automated verification. Additional analysis techniques for follow on analysis will be discussed in Chapter 5 as follow on work.

To automate analysis or the creation of viewpoints, this model uses multiple step traces and logic to query data from the model. Cameo System's Modeler enables data

queries through the use of several modeling languages that have access to the tool's application programming interfaces (API) or through structured expressions. This model used structured expressions, but a valued team member on most Cameo/SysML projects is a programmer with experience with the applications APIs to be able to pull advanced data quickly. Structured expressions within the tool allow for a combination of techniques to query data, as seen in Figure 47.

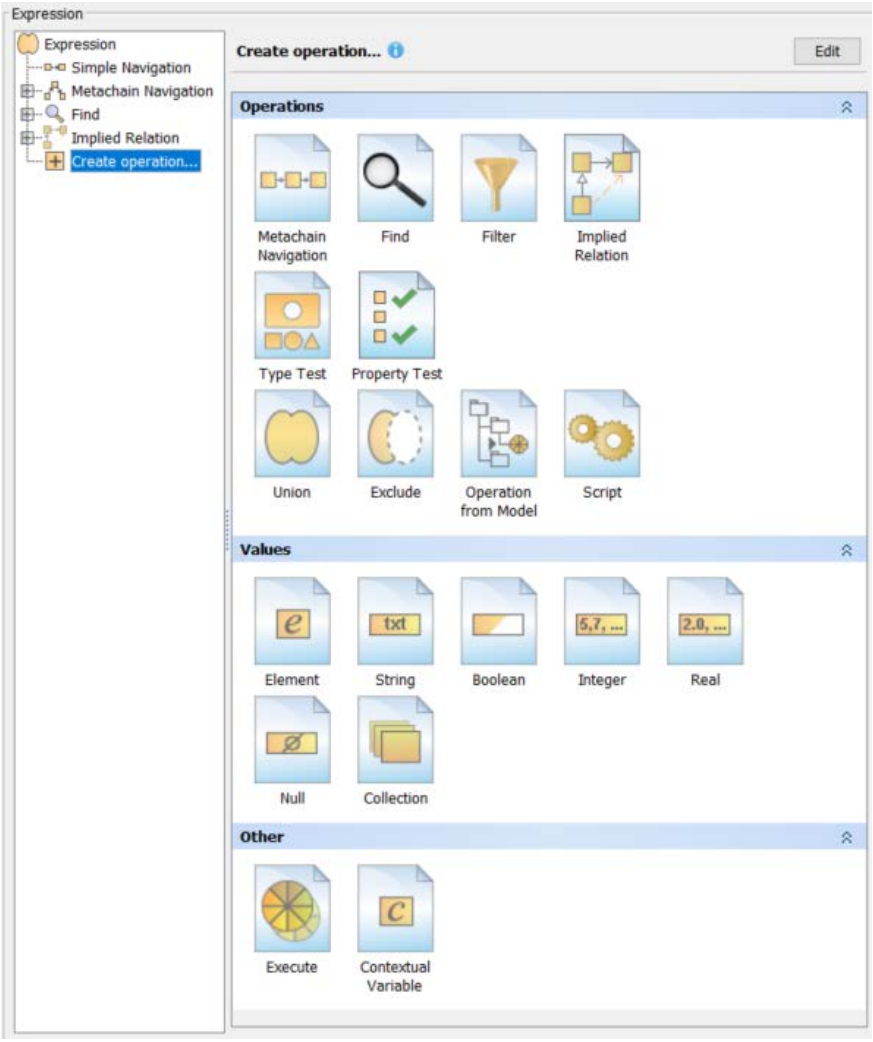

Figure 47. Cameo Structured Expression

The project evaluated the model for consistency against the guidelines that were initially set in the planning activity. The Cameo tool allows for the creation of validation scripts. This is an inappropriate term for them as they are truly used for verification of

requirements and design per INCOSE's definition (INCOSE 2017). An example for one of these scripts is provided in Figure 48 and Figure 49 for the following guideline: Actions within the logical definition of the framework shall be allocated to a part property of the logical WBS. Figure 48 shows the definition of the validation script which identifies the elements to be evaluated, Constrained Element set to CallBehaviorAction and the error message that will show for the elements that fail the script. Figure 49 shows a portion of the definition to identify the elements that do not meet the validation script. The structured expression identifies the parts of the WBS down to 3 levels, then identifies the actions allocated to that collection of parts, and then excludes that collection of actions from all actions. Table 11 shows an example output that is provided for this singular script for actions that at the time of running the script did not have an allocation relationship to the WBS. The validation scripts allow modelers to quickly identify gaps in the model and build confidence the model is built correctly.

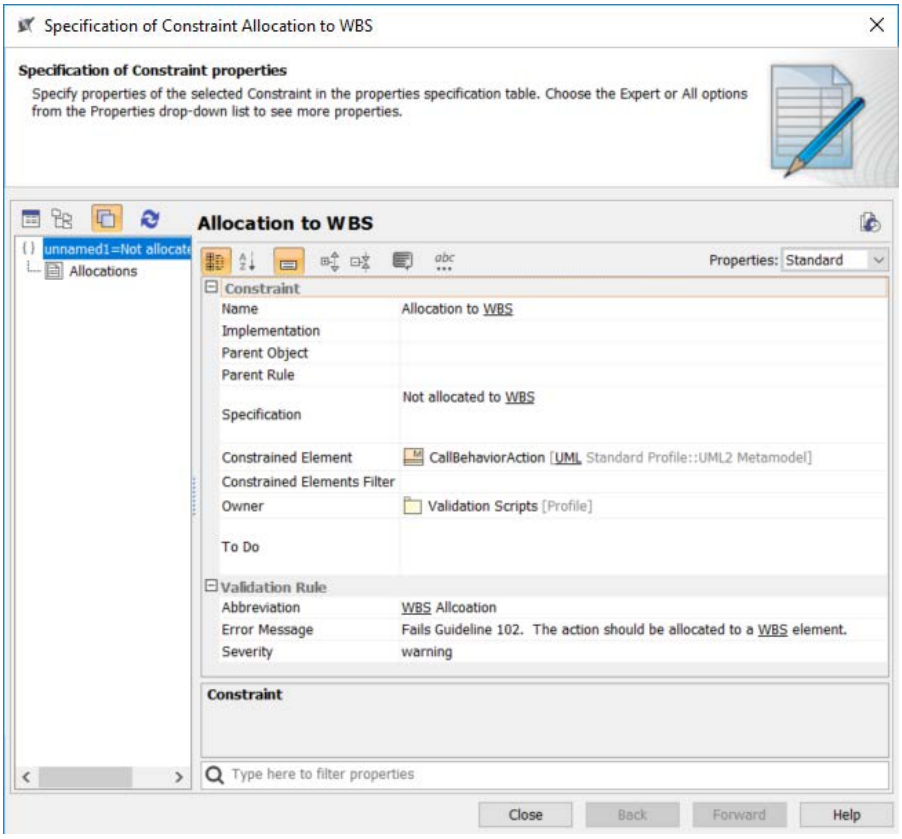

Figure 48. Validation Script Constraint

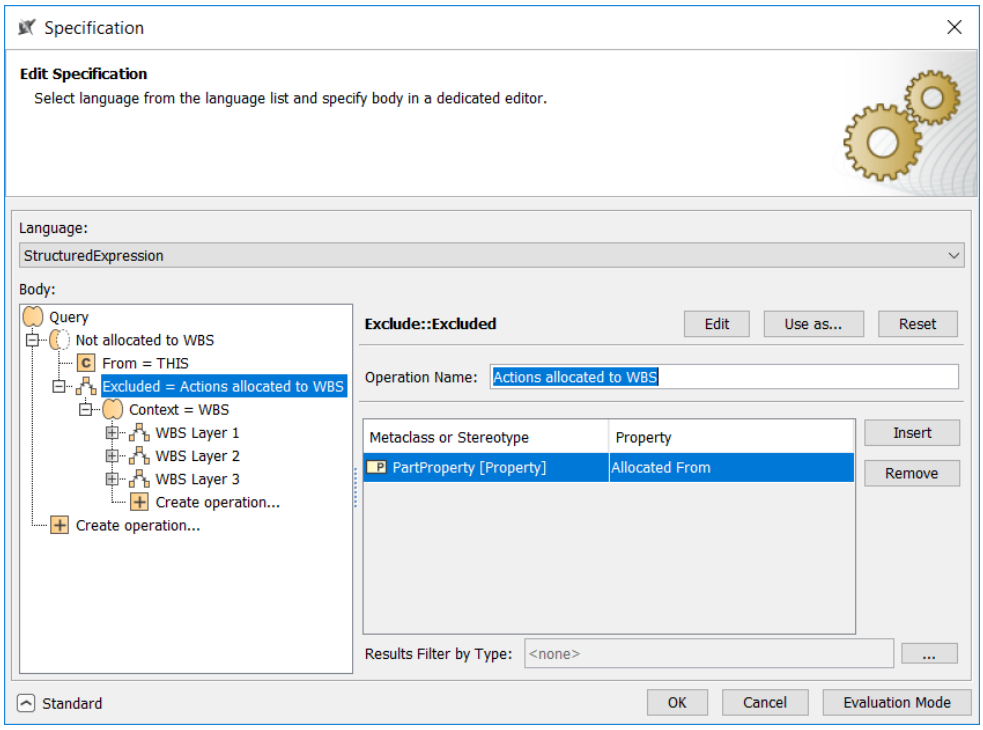

Figure 49. Structured Expression to Identified Actions Not Allocated to WBS

Table 11. Validation Script Output

Element	Severity	Abbreviation	Message
determine facility needs	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
develop data ontology	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
manage system architecture integration:Manage Architecture	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
plan classification marking	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
plan intellectual property protection	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
plan platform integration	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
review cdrls	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
review_oqe	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
schedule ws development	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.
store systems	warning	WBS Allcoation	Fails Guideline 102. The action should be allocated to a WBS element.

With verification of the model data through Cameo validation scripts, the model can use alternative views of the model data in tables and matrixes that query the model using the same techniques. This model contains several views with automated queries that help with manual review of the model. This capability allows readers of the model to see the same information from multiple viewpoints. Often multiple step traces were used to pull data into the tables. Figure 45 from Section IV.E is one example for how data can be queried. The relationships between the implementation and WBS are not direct relationships. The methodology shown in Figure 22 from Section III.E was used to build this matrix. Figure 50 shows the structured expression used to build query these relationships. Now, whenever a new allocation from either the implementation structure or the WBS is created to an action, the matrix will identify if a new abstraction relationship is also created between the WBS and implementation. For WBS parts that have allocations from more than one implementation part, the WBS could be decomposed further. The edge of this activity is when a process is co-chaired between two implementation parts such as a SETR event being chaired by both the program office and the prime contractor.

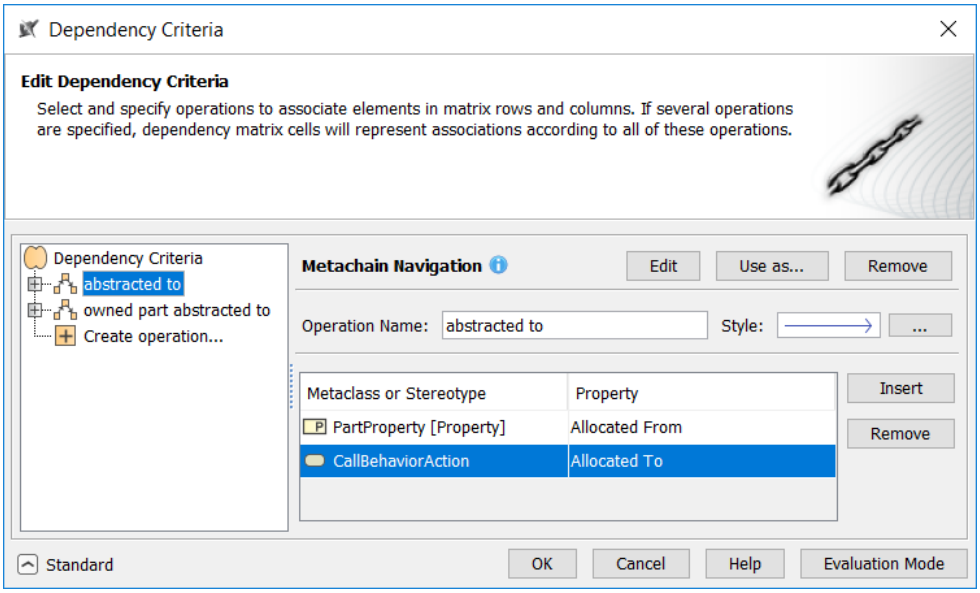

Figure 50. Structured Expression for Abstraction Relationship between Work Breakdown Structure and Implementation Structure

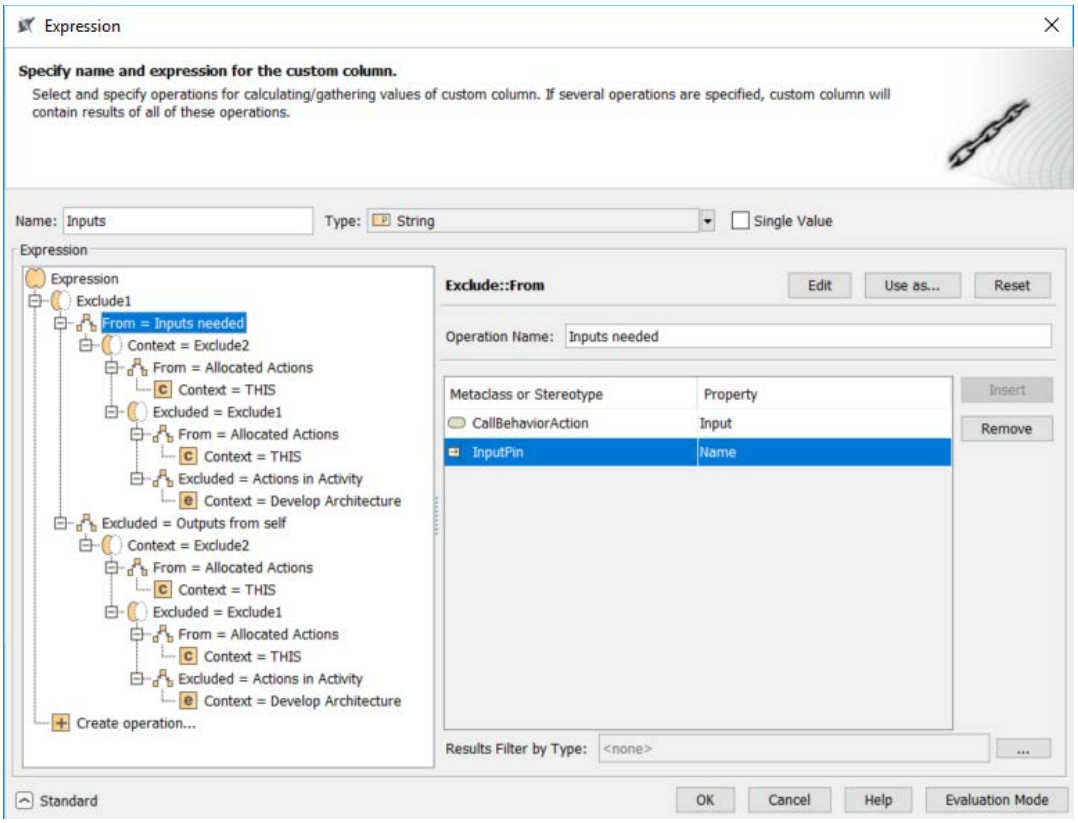

Figure 51. Structured Expression to Identify Interfaces between Work Breakdown Structure Parts

The project also used a diverse set of individuals with programmatic, systems engineering, MBSE, test, and network infrastructure experience. This project could have captured metrics for the model such as the number of types of SysML elements and completion of expected SysML relationships would be captured to increase confidence the project captured the MBSE Framework fully and correctly. Ideally, the MBSE Framework use for a real program and would be reviewed throughout the life of the program and updated at the SETR events as the SEP would be. Assessments for framework's effectiveness and lessons learned would be captured to improve the framework for future programs.

V. CONCLUSIONS

The first research question concerned the ability to model acquisition process. Research was conducted to review how acquisition processes are currently captured in paper-based products and also for how models are used to capture traditional systems. The processes to model a system were then modified to meet the needs of modeling an acquisition process.

The modeling processes was executed for an MTA Program up to a certain degree of fidelity to ascertain the benefits. Examples for how to capture traditional information within a process such as a WBS, responsibilities for program office vs contractor, and the information needs between them was shown. Manual built relationships between elements were shown in addition to examples for relationships that can be queried from the model.

The greatest benefit of modeling the acquisition process was found to the ability to look at the captured data from multiple views. Tables and matrices that are automatically updated after the queries are created were highly beneficial in understanding the content in the model. The model also allowed for the ability to reference traditional processes more specifically, or current program modeled processes more finely as opposed to the general method to reference an entire document that may be hundreds of pages. The model can also then be extended and referenced by others as they tailor the process to create an acquisition framework for a program's specific needs.

The second question concerned modeling MBSE methods within the process. MBSE processes were reviewed and the best practices identified were incorporated into the MTA acquisition process modeled. The ability to substitute new SysML Activities was found to be beneficial for modifying processes to incorporate MBSE methods. MBSE methods in general do not alter the overall traditional SE processes, but the more detailed functions below do need to be updated with different artifact inputs and outputs.

The model was also beneficial in defining the artifacts of an MBSE process. In traditional paper-based methods, some form of a document or drawing is the information being exchanged and the source of truth for information. In MBSE methods, the digital

artifacts that can be manipulated, queried, and/or executed should be the artifacts based. The model was beneficial in looking at a taxonomy and structure for these elements and then mapping this richer data to the processes used to create and input them.

Concerns with the process to capture the systems engineering process do exist as the toolsets for SysML are still immature. The current version of SysML relies heavily on classifier views of a system because it was based on UML (Object Management Group [OMG] 2017a). Stakeholders that need to review the model do not necessarily have the same level of understanding of SysML views that an experienced SysML modeler would. Stakeholders will likely not know the difference between a part property and a block or an action and an activity. These language and view limitations impede the distribution and benefits of the model. A new version of SysML, SysML 2.0, is under development that will greatly enhance SysML by clarifying the language use, expanding abilities of views, and formalizing analysis of architectures within the language (OMG 2020). These new capabilities will enhance the ability to benefit modeling activities by increasing readability of a model distributed to a wider audience. The process, examples, and metamodel presented in this thesis should be reevaluated once SysML 2.0 officially releases.

The process in this thesis allows for methods to capture data for a systems engineering process, but it did not capture simulated analysis on the process. The behavior modeled in activity diagrams, state machine diagrams, and sequence diagrams should be simulated with appropriate parameters to trade different behaviors. An example would be a function to create automated analysis of a model against guidelines vs manually reviewing the model. Both have benefits and increase confidence in the model. Automation will require upfront work that the program would hope to have a return on investment later in the life cycle. A manual review will increase confidence either way, but there are likely limits in the benefits of increasing the periodicity of manual reviews. This is similar to processes for reviewing software code with automation having great benefits, but manual code reviews at a certain coverage and periodicity providing value.

When looking at the entire life cycle, analysis can look at the much larger picture. With an abstract model, many large functions could be traded such as the number of flight tests, the percentage of tactical components in a HWIL system, or the types of digital or

ground verification events will be run. Will the program do a virtual reality and physical mockup of the system, or only do one. Each of these will impact the program's cost and schedule, but they will also increase confidence that an integration event will be successful. Using probabilities for actions to be successful, stochastic simulations could evaluate the benefits of adding MBSE methods or other activities into the life cycle. Stepanchick provides a good basis then for how one would evaluate a model based on a diverse set of options and looking for those with the greatest benefit to cost (2016).

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Department of Defense. 2007. "*The Defense Acquisition System.* DoD Directive 5000.01." Washington, DC: Department of Defense.

———. 2017. "*Operation of The Adaptive Acquisition Framework.* DoD Instruction 5000.02." Washington, DC: Department of Defense. Retrieved from http://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500002_dodi_2015.pdf.

———. 2018a. "*Defense Acquisition Guidebook*." February 26. Washington, DC: Department of Defense. Retrieved from https://www.dau.mil/tools/dag.

———. 2018b. "*Work Breakdown Structures for Defense Materiel Items*. MIL-STD-881D." Washington, DC: Department of Defense. Retrieved from http://assist.dla.mil.

———. 2018c. "*Standard Practice: Technical Data Packages*. MIL-STD-31000B." Washington, DC: Department of Defense. Retrieved from http://assist.dla.mil.

———. 2019. "*Operation of the Middle Tier of Acquisition (MTA).* DoD Instruction 5000.80." Washington, DC: Department of Defense. Retrieved from https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500080p.PDF?ver=2019-12-30-095246-043.

Estefan, Jeff A. 2007. "*Survey of Model-Based Systems Engineering (MBSE) Methodologies*." INCOSE MBSE Focus Group 25, no. 8.

Friedenthal, Sanford, Rick Steiner, and Alan Moore. 2014. *A Practical Guide to SysML*, 3rd ed. Burlington, MA: Morgan Kaufmann.

Grosklags, Paul. "*Outpacing the Competition: A Systems Engineering Challenge*". Keynote Speaker, "20th Annual NDIA Systems Engineering Conference," September 24, 2017, 21. Retrieved from https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2017/systems/Tuesday/Keynote_Grosklags.pdf.

Haskins, Bill, Jonette Stecklein, Brandon Dick, Gregory Moroney, Randy Lovell, and James Dabney. 2014. *Error Cost Escalation Through the Project Life Cycle*. *INCOSE International Symposium* 14(1): 1723–1737. Retrieved from https://doi.org/10.1002/j.2334-5837.2004.tb00608.x.

International Council on Systems Engineering. 2015. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th ed. edited by David Walden, Garry Roedler, Kevin Forsberg, Douglas Hamelin, and Thomas Shortell. Hoboken: Wiley.

NDIA Systems Engineering Division M&S Committee. (2011). Final Report of the Model Based Engineering (MBE) Subcommittee. Retrieved from https://www.ndia.org/-/media/sites/ndia/meetings-and-events/3187-sullivan/divisions/systems-engineering/modeling-and-simulation/reports/model-based-engineering.ashx.

Object Management Group. 2017. OMG Systems Modeling Language (formal/2017-05-01). Retrieved from https://www.omg.org/spec/SysML/1.5/PDF.

———. 2020. OMG Systems Modeling Language (SysML): Part 2 – System-Specific Metamodel and Libraries. ad/2020-07-02.

Office of the Deputy Assistant Security of Defense for Systems Engineering ODASD(SE). (n.d.). DoD Systems Engineering - Initiatives. Retrieved January 15, 2018, from https://www.acq.osd.mil/se/initiatives/init_de.html.

Stepanchick, Justin. 2016. “Integrating Model Based Engineering and Trade Space Exploration into naval acquisitions” Master’s thesis, Massachusetts Institute of Technology. http://dspace.mit.edu/handle/1721.1/104297.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California